MICROCOMPUTADOR CURSO PRÁTICO

TIRE O MÁXIMO DO MICRO EM 30 SEMANAS

PUBLICAÇÃO SEMANAL ILUSTRADA

29

Cr$ 12.900

Robôs em kits • Micros portáteis
Babel informática

rioGráfica

MANAUS, ALTAMIRA, MACAPÁ, JI-PARANÁ, PORTO VELHO, RIO BRANCO, SANTARÉM, BOA VISTA, SINOP, ALTA FLORESTA: Cr$ 16.770

MICROCOMPUTADOR CURSO PRÁTICO

APLICAÇÕES

Robôs para montar 577

Reconhecimento visual 589

SOFTWARE

Geradores de programas 580

LINGUAGENS

Múltipla escolha 583

Babel informática 592

RAIO X

Tandy Model 100 586

PERFIL

Medidata 596

TESTE

Depois de ler com atenção todo o fascículo, responda a estas perguntas. Veja a solução no próximo número, junto com um novo teste.

1)Qual o supermini brasileiro que utiliza a tecnologia do VAX-11/750?

2)Um sistema de reconhecimento de imagens que procura classificar o que "vê" segundo modelos pré-armazenados está utilizando uma abordagem "de cima para baixo" ou "de baixo para cima"?

3)Que linguagem tem seu uso e desenvolvimento supervisionado por uma comissão permanente?

4)Que motivos têm atraído os fabricantes de software à linguagem C?

RESPOSTAS DO TESTE ANTERIOR

1) Para que um Apple rode com o sistema operacional CP/M, utiliza-se a placa Z80, ou Softcard, que contém um processador Z80, em conjunto com o disco do sistema operacional.

2) Pode-se fazer cópias de segurança de arquivos em disco rígido em fita de videocassete.

3) O micro que baseou sua campanha de lançamento em imagens inspiradas no romance "1984", de George Orwell, foi o Macintosh, da Apple.

4) Um sistema especialista necessita, basicamente, de um banco de conhecimentos e um banco de regras para inferências e deduções.

PLANO DA OBRA

MICROCOMPUTADOR - CURSO PRÁTICO é uma coleção de 30 fascículos de periodicidade semanal. Os 29 primeiros terão, cada um, vinte páginas internas (miolo) e quatro capas, além de uma introdução de dezesseis páginas acompanhando o fascículo 1. O fascículo 30 incluirá o índice geral da obra. Os miolos, encadernados, constituem dois volumes de um curso prático de microcomputação, com seções indicadas por tarjas de diferentes cores.
As duas primeiras capas são descartáveis; as 3.as e 4.as trazem programas de jogos. Encadernadas, formam um volume de cem páginas.

COMO ENCADERNAR

As capas duras, incluindo as guardas, estarão à venda simultaneamente com os fascículos 10 (**Volume 1**) e 30 (**Volume 2** e **Jogos**).
Volume 1 - Deve ser encadernado com os elementos dispostos nesta ordem: guardas; frontispício e introdução (publicados com o fascículo 1); miolos dos fascículos de 1 a 15; e guardas.
Volume 2 - Guardas; frontispício (que virá com o fascículo 30); miolos dos fascículos de 16 a 30; e guardas.
Jogos - Guardas; frontispício e sumário (que constituem as 3.as e 4.as capas do fascículo 30); 3.as e 4.as capas dos fascículos de 1 a 29; e guardas.

rioGráfica **CONSELHO DE ADMINISTRAÇÃO:** Roberto Irineu Marinho; João Roberto Marinho; José Roberto Marinho; Oscar Neves; **Diretoria:** Oscar Neves; Danilo Esteves Costa; João J. Noro; Marcos March; Nilo Sérgio de Almeida; **GRUPO EDUCAÇÃO E CULTURA - Divisão Comercial - Diretor:** Jaime Rodrigues; **Gerente de Planejamento:** João de Azeredo Coutinho; **Gerente de Vendas:** Rubens Barbosa; **Coordenadores de Promoção:** Paulo Cesar M. Seixas; Edgard Mello M. Neto; **Gerente de Circulação:** Norberto Martin; **Supervisor de Operações:** Abel Pereira Pinto; **Editorial - Diretor:** João J. Noro; **Editor-chefe:** Maurício Rittner; **Diretor de Arte:** Adalberto Cornavaca; **Administração Editorial:** Heitor de Souza Paixão; **Editor:** Jesse Navarro; **Editor-assistente:** Claudio Marcondes; **Coordenadora de Revisão:** Isis Augusta Loyolla; **Revisores:** Cacilda Guerra; Carlos Eduardo Russo Tranjan; Maria da Graça Mendonça Couto; Sérgio Roberto Torres; **Chefe de Arte:** Bonifácio Duardes Miranda; **Chefe de Estúdio:** José Yuji Kuribayashi; **Assistentes de Arte:** Miguel Luis Escámez Simón; Edemilson Ribeiro da Silva; Wagner Celestino; **Colaboradores - Editores-assistentes:** Isa Mara Lando; Luiz Carlos Pizarro Marin; **Texto:** Carlos Eduardo Silveira Matos; José Antônio Arantes; Jussara Maturo; Paulo Brito; Solange A. Menezes; **Preparação de Texto:** Luiz Carlos Cardoso; **Tradução:** Luiz Roberto de Godoy Vidal; Maria Clara Cescato; Maria Ester Menezes Martinho; Fernando Dorfman Knijnik; Newton Roberval Eichemberg; **Pesquisa:** Stella Maria Quentel; **Assessoria Técnica:** Geraldo Coen; Gustavo José Ferreira Grubba; José Rubens Salles Toledo; Lauro de Lauro; Pierluigi Piazzi; **Estúdio:** Vagner Facuri de Oliveira. As fotos não creditadas pertencem à obra original; Medidata.

© APSIF Copenhagen, 1984; © Orbis Publishing Ltd., 1984; © Editora Rio Gráfica Ltda., 1985, para a língua portuguesa.

Composição: Editora Rio Gráfica Ltda. e AM Produções Gráficas Ltda. **Impressão:** JBIG.

NÚMEROS ATRASADOS - Você poderá comprar os exemplares que faltam em sua coleção, pelo preço do último fascículo posto à venda, em sua banca de jornal preferida. Caso não consiga obtê-los, faça sua solicitação por carta endereçada diretamente à Editora Rio Gráfica Ltda., rua Itapiru, 1209, Rio Comprido, Rio de Janeiro, CEP 20251. O atendimento será feito por via postal. Nas cidades do Rio de Janeiro e de São Paulo, as compras poderão ser efetuadas pessoalmente nos seguintes endereços: Rio de Janeiro - Rua Itapiru, 1209, Rio Comprido; São Paulo - Rua Frei Caneca, 1152, Consolação.

Distribuidor exclusivo para todo o Brasil: Fernando Chinaglia - Rua Teodoro da Silva, 907, Rio de Janeiro. Distribuidor para Portugal: Electroliber Lda. - Rua Prof. Reinaldo dos Santos, 1488, Lisboa.

ROBÔS PARA MONTAR

Os kits robóticos nem sempre propiciam passatempo satisfatório. Pensando nisso, a Fischertechnik produziu um pacote que permite ao usuário elaborar seus próprios projetos, além dos seis que o acompanham.

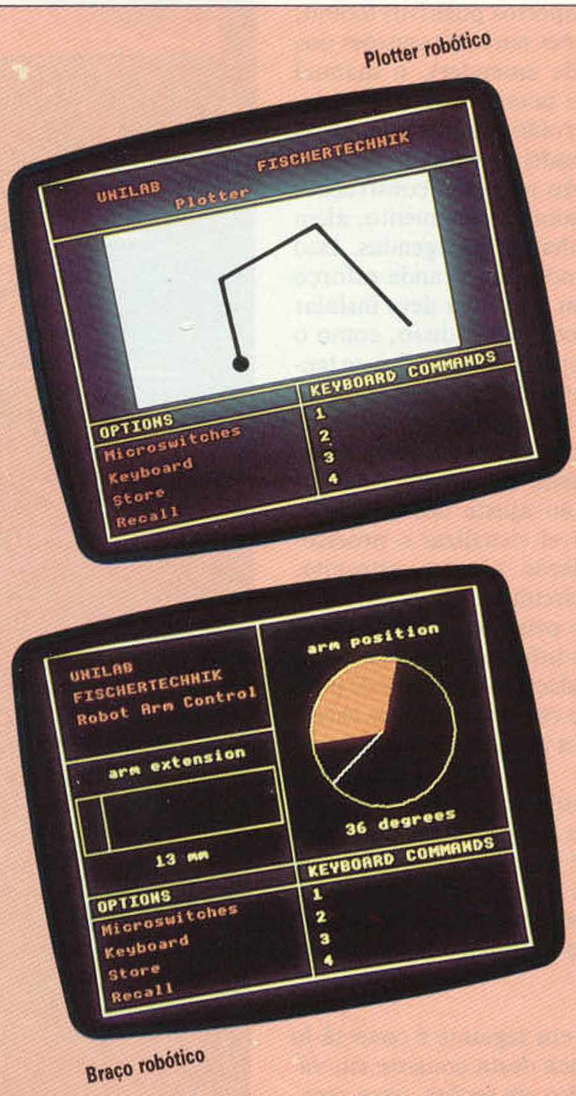
Embora profissionais da indústria de microcomputadores tenham previsto um grande desenvolvimento no campo da "robótica doméstica", ou seja, dos robôs para o lar, até o momento isso ainda não ocorreu. As razões são mais ou menos óbvias. Apesar dos inúmeros robôs domésticos existentes no mercado, todos ainda apresentam consideráveis imperfeições, deixando o público em dúvida sobre se de fato vale a pena iniciar-se na robótica.

Existem, no mercado internacional, robôs de fácil montagem, como os Movits, que pessoas sem conhecimento de robótica ou eletrônica conseguem montar em poucas horas. O problema desses dispositivos é terem aplicações extremamente limitadas, fazendo o usuário, após algum tempo, perder o interesse por eles. O mercado oferece também robôs mais caros, cuja operação requer em geral amplas noções de robótica e eletrônica. Muitos desses robôs, porém, têm uma única aplicação. Isso significa que, quando todas as possibilidades foram exploradas, o robô é posto de lado. Assim, parece não haver dúvida de que seria desejável um robô de fácil montagem e capaz de executar várias tarefas.

A isso se propôs a Fischertechnik Robotics Kit, baseando seus kits numa idéia bastante simples. Há anos encontram-se no exterior diversos conjuntos eletrônicos para montar, destinados a crianças, que possibilitam a construção de circuitos simples, como, por exemplo, um rádio elementar. Pode-se desmontar e remontar esse mesmo kit, de modo a formar um timer ou um alarme anti-roubo. Muitos de nós brincamos em criança com jogos de armar, que permitem a construção de inúmeros modelos diferentes. A Fischertechnik, ao projetar seus kits robóticos, associou essa idéia à moderna tecnologia de computadores. Como resultado, desenvolveu um produto que poderá significar para a robótica o avanço e difusão que tantos vêm anunciando.

O Kit Robotics oferece ao usuário a possibilidade de construir diferentes dispositivos robóticos acionáveis por microcomputador: desde um braço mecânico e uma máquina que ordena objetos de vários comprimentos, até um plotter e um dispositivo para entrada de gráficos. Após realizar suas experiências, o usuário tem a opção de desmontar o robô para construir outro. O controle por computador requer uma interface, para transformar os sinais digitais do computador em sinais que poderão ser usados na operação dos motores elétricos do kit.

Como as peças são de material plástico, podem-se encaixar os componentes tal como num jogo de armar comum. Blocos ligados formam as partes mais longas dos braços ou servem para sustentar as unidades de controle. Outras peças destinadas à construção das unidades mecânicas são as engrenagens e os parafusos sem fim que as fazem girar. Há também uma grande placa de plástico, medindo 260 x 187 mm, usada como base para o robô — ou para o papel, caso se incorpore um plotter ou um tablete gráfico. O controle elétrico e a alimentação também



Monitorando o movimento

Ao lado, vêem-se as telas iniciais do software da interface. A foto de baixo mostra o programa que controla o braço mecânico. À direita da tela, um diagrama circular registra o movimento horizontal. À medida que o braço deixa a posição zero e vai para a direita, a linha amarela se move no sentido horário, e o número de graus indicado embaixo aumenta. Com a extensão do braço, a linha dentro da barra (na parte esquerda do vídeo) e os milímetros mudam conforme o movimento "real". Na parte inferior da tela, estão as opções de comando disponíveis controladas por microinterruptores da placa ou pelo teclado. As seqüências de movimentos armazenadas no compu.ador podem ser chamadas e executadas de novo.

Na tela superior, temos o mesmo sistema de controle com o plotter robótico, com a diferença de que se acrescentam comandos extras para levantar e baixar a caneta sobre o papel. À medida que o plotter desliza pelo papel, os movimentos são espelhados na tela.

requerem montagem de componentes. Os kits incluem vários conectores macho e fêmea, oito interruptores e dois motores elétricos. Acompanham ainda cerca de 1 m de cabo flexível de vinte vias e um pequeno cabo suplementar, para emendas. As peças são robustas e dão impressão de grande durabilidade. A única exceção é a base do potenciômetro, que se quebra com relativa facilidade.

O potenciômetro apresenta uma pequena desvantagem: ao contrário do restante da instalação elétrica, cuja fiação pode ser parafusada nos conectores, a única maneira de se ligarem os fios desencapados ao potenciômetro é enrolando-os em volta dos conectores, e soldando-os, a fim de evitar curtos-circuitos, que danificariam o potenciômetro.

Ajuste dos componentes

A construção desses robôs não exige habilidade especial. A maior parte de suas peças é bem resistente, o suficiente para que mesmo dedos inexperientes consigam manejá-las com sucesso. O maior problema envolvido na montagem está no próprio manual — talvez a parte mais fraca do conjunto. Ao lado de aspectos positivos (como, por exemplo, as instruções para as conexões dos circuitos, bem fáceis de entender), o manual baseia-se em ilustrações pouco claras de modelos parcialmente construídos. Fotos das peças que compõem determinado modelo são mostradas junto com outras do robô em construção, mas ressentem-se de maior detalhamento, além de não virem acompanhadas de legendas. Isso faz o construtor despender um grande esforço para tentar descobrir em que lugar deve instalar determinado componente. Além disso, como o robô é mostrado apenas num ângulo, fica-se tentando adivinhar onde encaixar certa peça quando ela não aparece na ilustração.

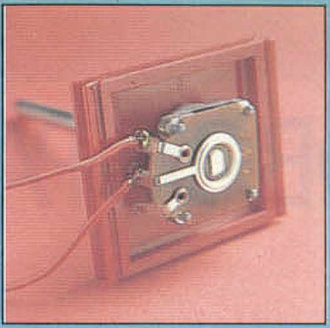
Há um outro agravante: as fotos são em branco e preto. Então, quando se chega à etapa final, que corresponde ao ajuste dos circuitos, torna-se quase impossível visualizar o procedimento correto de instalação. Um fato estranho, considerando-se que o circuito depende em grande parte da codificação pelas cores do cabo flexível. A quantidade de informações fornecidas pelos diagramas de instalação é mínima e de pouca ajuda; simplesmente estabelecem, por exemplo, que os fios de E3 a E8 pertencem a determinada seção.

Apesar disso, devemos esclarecer que o robô mostrado na foto deste artigo foi montado por uma pessoa totalmente leiga em robótica e eletrônica. Mesmo assim, seria conveniente que o fabricante não confiasse tanto no bom-senso das pessoas e melhorasse o manual.

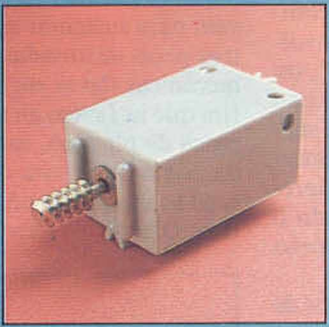
Mãos à obra

Montado o robô, a tarefa seguinte é conectá-lo a uma interface. A placa desta consiste em vários conectores, divididos em seções, para prender os fios condutores do cabo flexível. A inter-

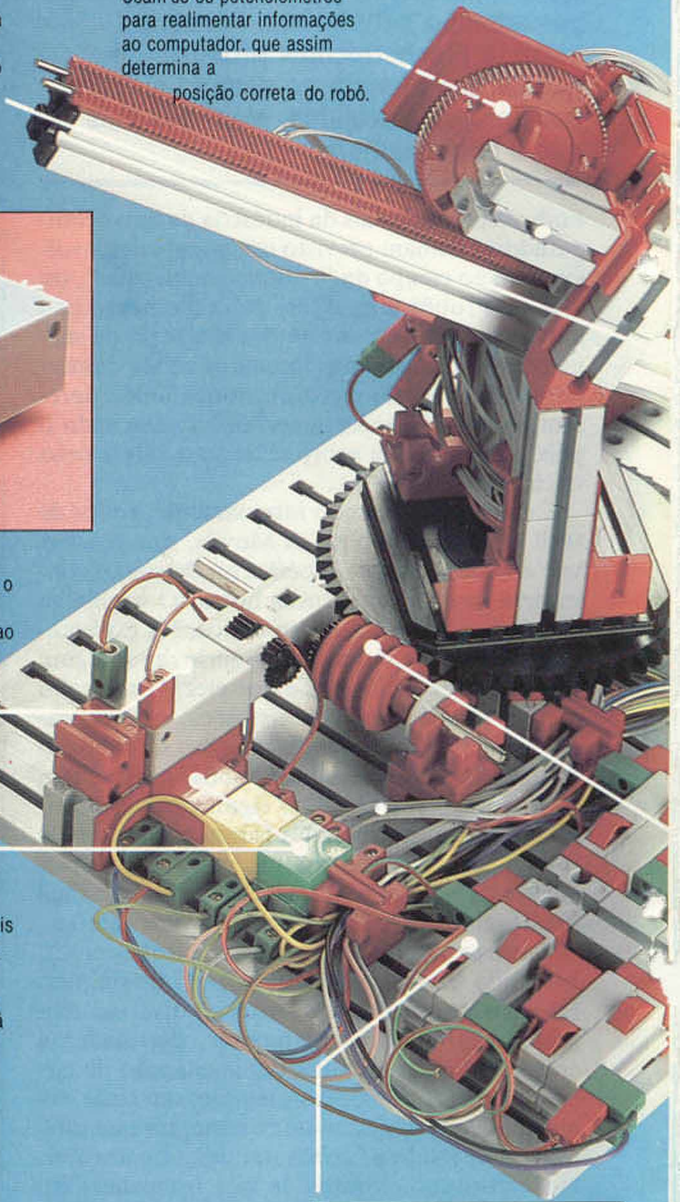
Eletroimã
A barra de metal no topo do eletroimã encaixa-se na garra do braço. Pode-se ligar e desligar o imã, possibilitando ao robô levantar pequenas placas de metal fornecidas com o kit.



Potenciômetro
Usam-se os potenciômetros para realimentar informações ao computador, que assim determina a posição correta do robô.

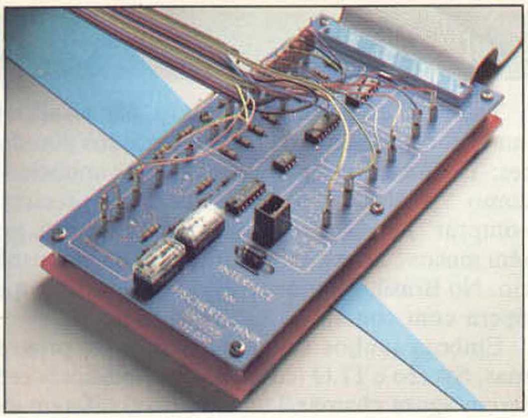


Motor
Uma caixa de plástico abriga o motor, conectado por duas saídas. O parafuso sem fim, ao girar, impulsiona as engrenagens que movem os robôs.

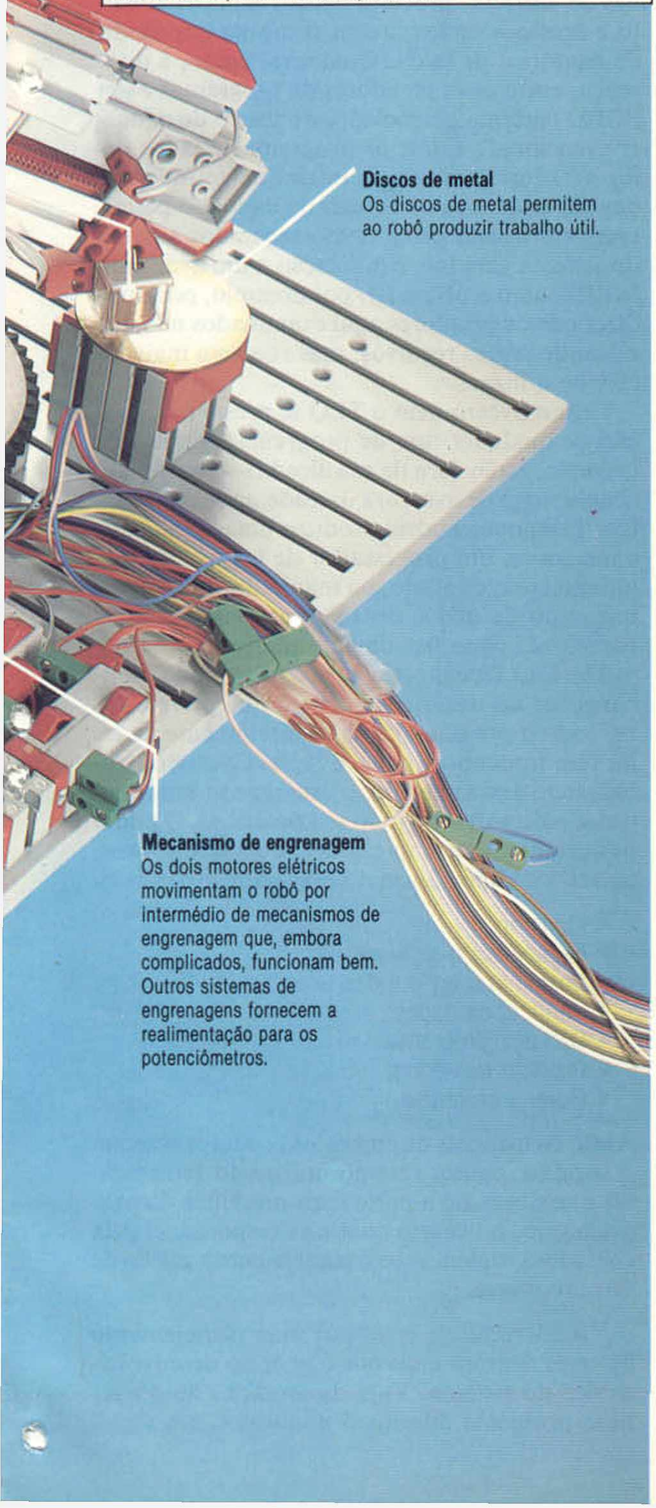


Lâmpadas
Sempre que um dos três sistemas de controle principais (imã, motores de movimentos horizontal ou vertical) é acionado, uma delas acende, informando qual controle está sendo usado.

Interruptor
Há oito, que correspondem às oito linhas de dados no conector de expansões do micro. Combinados, geram um número no registrador de direção de dados; este orienta os movimentos do robô.



Força para o robô
Esta interface permite que se controle um kit robótico por intermédio de um micro. Os fios do robô encaixam-se nos conectores da interface, que, por sua vez, liga-se ao micro pelas saídas das expansões e da impressora.



Discos de metal
Os discos de metal permitem ao robô produzir trabalho útil.

Mecanismo de engrenagem
Os dois motores elétricos movimentam o robô por intermédio de mecanismos de engrenagem que, embora complicados, funcionam bem. Outros sistemas de engrenagens fornecem a realimentação para os potenciômetros.

face mostrada na foto é a do BBC Micro, embora as de outros computadores talvez não difiram muito. No canto direito da placa estão as entradas para interruptor. Sob elas estão os dois potenciômetros, que possibilitam ao computador monitorar a energia fornecida aos interruptores e às lâmpadas elétricas. A saída analógica mede o controle dos potenciômetros.

No lado contrário da placa de interface encontram-se as conexões que controlam os motores e um eletroímã (usado pelo braço mecânico para apanhar objetos de metal). Finalmente, há o conector da fonte de energia (6 a 8 V), que alimenta todo o sistema.

A interface conecta-se ao BBC Micro por três saídas separadas (o conector de expansões, a saída analógica e a da impressora), utilizadas para controle dos motores e do eletroímã. Como o sistema se destina a usuários inexperientes, construiu-se a placa de interface prevendo-se proteção em caso de ligações incorretas, que poderiam provocar voltagens elevadas nas interfaces do computador e eventualmente danificá-lo.

A linguagem Prof

Juntamente com a interface, o kit traz um cassete com três programas que auxiliam no controle do robô. Os dois primeiros, especializados, controlam o braço e o plotter, permitindo aos dispositivos externos o controle pelo teclado ou pelos interruptores do próprio robô. O terceiro programa, o Prof (abreviação de professor), é um sistema operacional genérico, através do qual comanda-se o robô diretamente. Trata-se de uma extensão do BASIC, acrescido de alguns comandos. Muitas das palavras adicionais introduzidas pelo Prof são variáveis — como Motor e Magnet (ímã) — para os endereços de controle, enquanto outras servem para auxiliar a programação das seqüências de movimentos que os robôs deverão realizar.

O Prof, sem dúvida, é uma linguagem com muitos recursos, permitindo loops condicionais e programação estruturada, mas confina-se principalmente ao controle de saída. Assim, não dá suporte ao painel gráfico, um dispositivo de entrada, porque a linguagem não armazena informações provenientes de dispositivos de entrada. Por conseguinte, para tais fins, o usuário deverá criar seus próprios programas em BASIC.

Apesar dos problemas com a documentação, o kit e a interface da Robotics constituem um sistema razoável. Os robôs têm um desempenho satisfatório e podem ser construídos em poucas horas. O usuário não precisa se ater apenas aos seis projetos que acompanham o kit. Uma vez familiarizado com o sistema, poderá construir inúmeros modelos.

O kit não é um brinquedo barato e o manual tem os seus defeitos. Mas, uma vez superadas essas dificuldades, o kit e a interface da Fischertechnik Robotics irão firmar-se como uma iniciação ideal para o usuário interessado em explorar o campo da robótica.

GERADORES DE PROGRAMAS

O usuário já pode desenvolver seus próprios aplicativos. Com os geradores de programas, instruções genéricas em linguagem muito próxima da comum são passadas ao BASIC sem erros.

É possível um software que elimina o trabalho árduo e maçante da programação, produzindo seqüências de instruções em BASIC sem erros de sintaxe? Nem sonho nem ficção científica, os geradores de programas são ferramentas já em uso por programadores e software houses. Eles transformam instruções genéricas, emitidas pelo usuário em linguagem próxima da comum, em instruções em BASIC, com sintaxe correta e rotinas de erro incorporadas.

Essas características estão presentes, em algum grau, em linguagens como LOGO e FORTH, que não aceitam a introdução de palavras estranhas ao código ou não definidas pelo usuário. Contudo, esses recursos "inteligentes" de pouco valerão se quisermos, por exemplo, programar o cálculo de nosso imposto de renda.

Hoje, porém, a multiplicação de programas incorporando conceitos de inteligência artificial torna menos remota a possibilidade de um usuário leigo criar aplicações sob medida com um mínimo de esforço.

O Microtext, por exemplo, um software desenvolvido na Inglaterra, gera questionários, sistemas de informação e programas educacionais. Como mostra a ilustração, Microtext pode criar programas que instruem um usuário leigo a executar complexas tarefas técnicas. No modo de autoria, o Microtext permite testes e alterações; finalizando o programa, o software transforma-o num sistema pronto para execução, que o usuário não pode mais alterar. Disponível no mercado internacional, o Microtext roda em CP/M.

Outro programa bastante sofisticado é o The Quill, dedicado a gerar jogos de aventura. Ele permite montar um banco de dados de informações — lugares, objetos, como manipulá-los etc. — e incorpora todos esses dados em linguagem ASSEMBLY, produzindo um programa que roda sem erros desde a primeira execução. Um acréscimo útil é o programa Illustrator, que gera gráficos para as aventuras do The Quill.

Este e o Illustrator foram desenvolvidos para rodar em máquinas de recursos modestos, como o Sinclair Spectrum e o Commodore 64. Apesar do baixo preço, o The Quill já foi utilizado por software houses ingleses para produzir diversos jogos de aventuras lançados comercialmente.

The Quill e Microtext, contudo, não rivalizam em capacidade com os mais avançados geradores: The Last One (pretensiosamente anunciado como "o último software que você precisará comprar") e o Sycero, ainda mais poderoso, porém menos "amistoso" na interação com o usuário. No Brasil, foi desenvolvido o Gensoft I, que opera com comandos em português.

Embora ambos sejam geradores de programas, Sycero e TLO (como os distribuidores preferem agora chamar The Last One) diferem em concepção, abrangência e na maneira de definir as necessidades do usuário. Ambos geram programas em BASIC, e são extremamente eficientes na manipulação de dados, incluindo as rotinas de entrada, processamento, armazenamento e acesso a dados, assim como na impressão de relatórios de saída. Qual será, então, a diferença, entre esses geradores de programas e um SGBD (sistema gerenciador de banco de dados) convencional? É que os programas criados pelos geradores tornam-se inteiramente independentes, enquanto os bancos de dados só permitem desenvolver programas executados a partir do sistema gerador. Em relação a um avançado SGBD como o dBase III, por exemplo, podemos dizer que os geradores aqui examinados não dispõem de tantos recursos, mas são bem mais fáceis de utilizar.

Nem o Sycero nem o TLO se prestam à geração de qualquer tipo de programa. Jogos, por exemplo, estão fora de sua alçada — mesmo um simples jogo de aventura baseado apenas em textos. Tampouco podem produzir uma planilha financeira ou um processador de textos. Seu valor revela-se, contudo, na manipulação de qualquer tipo de dado, do cálculo de impostos às complexas equações de engenharia.

The Last One mostra-se mais adequado às inclinações do usuário moderno, que prefere fazer todo o pré-planejamento diretamente na tela, sem rascunhos, papel e lápis. Contudo, trabalhando dessa maneira, o usuário não estará extraindo do software todo seu potencial. O plano de ação do Sycero, que consiste em sete pontos, enfatiza a importância do planejamento prévio:

- Planejar;
- Especificar o sistema;
- Desenhar o lay-out das telas;
- Verificar os dados;
- Definir o programa;
- Produzir relatórios;
- Gerar o programa.

Aliás, os manuais de ambos os produtos realçam o seguinte ponto: mesmo utilizando ferramentas que eliminam a parte mais mecânica da programação, o usuário continua responsável pela cuidadosa organização e planejamento prévio de seu programa.

"A correção de erros por mau planejamento às vezes demora mais que o próprio desenvolvimento do sistema. Fuja da tentação de pensar num problema durante 5 minutos e, em segui-

da, correr para colocar na máquina o disco do Sycero, construindo o sistema enquanto trabalha. Esse não é um bom método.

"Comece sempre fazendo uma lista detalhada de tudo o que você pretende que o computador produza, tanto dados mostrados na tela (informações on line) como relatórios escritos. Só depois de determinar com precisão o que deseja é que deve definir quais as informações que entrarão, a fim de obter tais resultados."

O manual do TLO também dá conselhos objetivos:

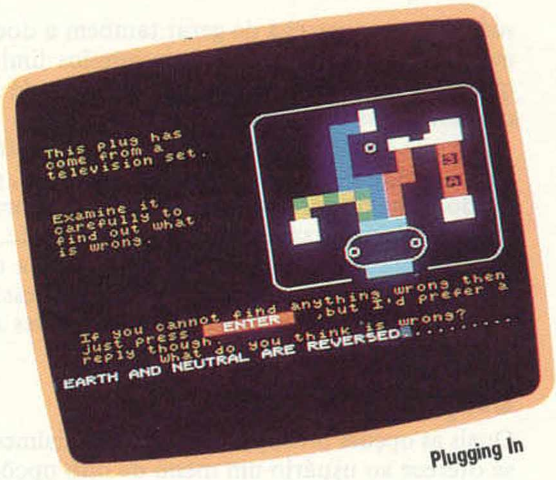
"Planeje o fluxo geral de seu programa e antecipe os erros que o usuário final irá cometer; prepare o programa, então, para deter esses erros com segurança. Na verdade, será melhor planejar tão detalhadamente quanto você faria com um programa comum. Para tanto, inicie elaborando diversos programas curtos, interligados por um simples menu, que chama cada um quando necessário. Você verá que esse sistema de construção modular permite criar com facilidade programas extensos e complexos".

Qualquer software que objetive possibilitar ao usuário novato a criação de suas próprias aplicações computacionais precisa satisfazer plenamente alguns quesitos. Portanto, deve:

- Encorajar a programação estruturada "de cima para baixo", na qual o usuário primeiro esboça os contornos gerais do programa e, depois, especifica cada ponto com mais detalhes, conforme dá seguimento ao trabalho;
- Ser dirigido por menus de múltipla escolha, com mensagens que orientem o usuário na escolha das opções disponíveis;
- Detectar erros assim que são introduzidos, e permitir correções e melhoramentos;
- Ser "transportável", isto é, capaz de rodar em outro equipamento, independente do sistema de software que o gerou;
- Produzir ampla documentação do processo de geração do programa, de modo que um novo usuário, mesmo desconhecendo as etapas de criação, possa introduzir melhorias ou correções.

Todas essas medidas auxiliam sobretudo o usuário novato. Contudo, os geradores de programas não são ferramentas restritas aos iniciantes; devem ser aceitáveis também para os mais experientes, não os enredando e atrasando com excesso de testes e sistemas de segurança.

Tanto o Sycero como o TLO operam por meio de menus informativos. Dos dois, o segundo é o que mais segue a abordagem estruturada. Ele começa produzindo um fluxograma, na verdade uma série de comentários esclarecedores sobre a finalidade de cada seção do programa. Por exemplo, Desvio para menu de quatro opções: esse comentário, se se desejar, pode ser incorporado ao programa final em BASIC sob a forma de linhas de REM. Definidas as entradas e as telas, o programa resultante em formato ASCII deverá ser gravado em código binário.



Plugging In

Este programa instrucional gerado pelo Microtext apresenta questões a respeito de sistemas elétricos. Aplicações educacionais, que envolvem interação com o usuário e comparação de suas respostas com dados pré-gravados, são ideais para os geradores de programas.

Já no Sycero, começa-se por definir os arquivos e o desenho das telas. Em seguida, vem o processamento das variáveis de entrada, ou seja, a associação de determinados dados de entrada às respectivas mensagens de erro ou de auxílio ao usuário. Definem-se também os relatórios de saída e outras opções. Os vários módulos são então interligados, imediatamente antes de serem codificados em linhas de programa.

Embora esse processo não seja tão fácil para o novato, tem a vantagem de exigir uma cuidadosa definição do sistema todo antes da fase de programação; caso contrário, o programa simplesmente não roda. Se forem detectados erros durante a codificação, o gerador interrompe o processo e informa qual o erro em questão, possibilitando ao usuário corrigi-lo. Ao fim da codificação, o Sycero automaticamente grava o programa num arquivo binário, que pode ser executado a partir do próprio gerador.

Tanto o Sycero como o TLO incluem complexas e abrangentes rotinas de detecção de erros nos programas que geram, aumentando consideravelmente seu tamanho em relação a um programa equivalente em BASIC.

Versão brasileira

No Brasil, a Gensoft Informática Ltda., com sede em São Paulo, desenvolveu o Gensoft I. Segundo os fabricantes, o Gensoft I alia um gerador de aplicativos a um gerenciador de banco de dados relacional. Isso permite que, com pouquíssimos comandos — todos em português —, o usuário crie seus programas, definidos de forma coloquial, sem necessidade de conhecimentos específicos de informática ou de linguagens computacionais. Basta definir, na tela, os relatórios de que se necessita, as informações a serem armazenadas e processadas, e determinar as relações, os testes e os cálculos com os dados dos arquivos. A rapidez de execução do Gensoft I é garantida por suas rotinas em ASSEMBLY. Compõe-se de oito módulos, sendo três de desenvolvimento (texto, formato e cálculo) e cinco de execução (arquivos, utilitários, tratamento, relatórios e menu do usuário). O próprio

Faça você mesmo
The Last One (TLO) e Sycero são softwares que permitem ao usuário criar programas em BASIC bem estruturados e livres de erros de sintaxe, e que são executados independentemente do software gerador. Contudo, para utilizá-los com proveito é necessário um cuidadoso planejamento.

software se encarrega de gerar também a documentação do sistema, que tem amplos limites quanto à quantidade e tamanho máximo de arquivos, registros e campos.

No que diz respeito à documentação, tanto o Sycero como o TLO nada deixam a desejar, gerando listas de características tais como nomes das variáveis utilizadas, formatos de tela etc. O Sycero acrescenta automaticamente a data e a hora, tanto em tela como em cópias impressas, de modo a facilitar a distinção entre versões antigas e mais recentes de um programa.

Etapas da criação

Quais as opções principais do TLO? Inicialmente se oferece ao usuário um menu de oito opções.

Criar programa	Consultas
Modificar programa	Formatar novo disco
Modificar arquivo	Continuar a codificação
Definir arquivo	Voltar ao BASIC

Depois de escolher a primeira opção e responder afirmativamente à pergunta Seu programa requer arquivos?, definem-se os arquivos necessários. Assim como o tamanho e o tipo dos campos. A tela apresentará então o Menu de criação e de fluxograma. Em sua última versão, o TLO apresentava vinte opções nesse ponto:

Listar fluxograma	Reinicializar
Modificar fluxograma	Posicionar ponteiros
Codificar programa	Ler arquivo
Unir fluxogramas	Gravar em arquivo
Interromper	Pesquisar ou classificar arquivo
Entrada pelo teclado	Unir programas
Exibir dados	Verificar registros
Desvios	Limpar arquivo
Cálculos	Funções de banco de dados
Funções especiais	Multifunções

Utilizando essas opções, elabora-se um fluxograma, a exemplo deste a seguir, que atualiza e pesquisa um arquivo de endereços:

1 ..Desvio para um menu de três opções
2 ..Posicionar ponteiro no fim do arquivo
3 ..Entrada pelo teclado
4 ..Gravar dados de entrada no arquivo
5 ..Perguntar <Terminou?>.Desviar se "Não"
6 ..Desvio incondicional direto
7 ..Desvio para um menu de três opções
8 ..Posicionar ponteiro no início do arquivo
9 ..Pesquisa do arquivo pelo teclado
10 ..Exibir dados do arquivo
11 ..Perguntar <Outra consulta?>.Desviar se "Sim"
12 ..Desvio incondicional direto
13 ..Classificar arquivo
14 ..Posicionar ponteiro no início do arquivo
15 ..Ler dados do arquivo
16 ..Exibir dados do arquivo
17 ..Desvio incondicional direto
18 ..Finalizar

Definida assim a estrutura do programa, o gerador pode iniciar a codificação. Até aqui, não se pode sair do TLO sem perder todo o trabalho feito. Iniciada a codificação, contudo, é possível interrompê-la a qualquer ponto, e reiniciá-la por meio da função do menu inicial Continuar a codificação. Nessa fase, definem-se o desenho das telas e os destinos dos desvios; no exemplo dado, o menu definido no ponto 1 irá desviar para 2 (gravar novos dados), 7 (pesquisar os dados do arquivo) ou 18 (finalizar).

Convém gravar em separado as telas definidas; o Sycero faz isso automaticamente, ao passo que, no TLO, requer-se um comando especial.

O menu de abertura do Sycero oferece treze opções, que serão utilizadas mais ou menos na ordem em que vêm listadas:

- Configuração do sistema
- Inicialização
- Definição dos arquivos e dos campos
- Definição das telas
- Processamento das telas
- Definição de relatórios
- Processamento dos relatórios
- Definição do programa
- Geração do programa
- Criação de arquivo de dados reais
- Execução do programa gerado
- Utilitários (programas de apoio)
- Finalizar a sessão

Seguindo esses passos, ao atingir a etapa Geração do programa, a maior parte do trabalho já estará pronta. As fases que se seguem requerem poucas intervenções do usuário, a não ser que o software detecte um erro básico. Em outras palavras, quanto mais detalhadas forem as fases iniciais de planejamento, mais rápida e simples será a criação do programa.

Os geradores de programas se assemelham aos compiladores, que transformam as instruções no BASIC ou outra linguagem de alto nível em comandos de código de máquina. Em ambos os casos, o software é uma ferramenta que traduz comandos genéricos, mais próximos ao usuário, para códigos mais próximos da máquina. Os geradores podem fazer essa conversão, que é um trabalho mecânico, mas não deduzem as intenções do usuário. O principal, portanto, consiste em instruir o sistema com a máxima precisão.

MÚLTIPLA ESCOLHA

Examinamos aqui os vários métodos de efetuar a subtração e a multiplicação em linguagem ASSEMBLY. Apresentamos também um novo tipo de operadores lógicos — os de deslocamento e rotação.

Tanto o Z80 como o 6502 admitem a instrução SBC ("subtração com transporte"), mas sua execução difere bastante em cada processador. O 6502 usa o flag de transporte para lidar com o recurso "empréstimo", que equivale ao transporte na adição. Na linguagem ASSEMBLY do Z80, a instrução SBC funciona exatamente como a ADC — o flag de transporte indica o resultado da operação.

Vamos supor que acrescentamos $E4 a $5F, usando ADC, tendo antes zerado o flag de transporte. O resultado no acumulador é $43, com o flag de transporte ativado, mostrando que o resultado verdadeiro é $0143. Houve um estouro para o flag de transporte, porque o tamanho do acumulador não basta para conter o resultado total.

Suponhamos, agora, ainda no Z80, que zeramos mais uma vez o flag de transporte e subtraímos $E4 de $5F: o resultado no acumulador será $7B e o transporte será ativado. Agora, se acrescentamos $7B a $E4 (tendo zerado mais uma vez o flag de transporte) verificaremos que o resultado no acumulador é $5F e que o flag de transporte encontra-se ativado. Isso está totalmente coerente, como demonstra

$5F − $E4 = $7B	transporte ativado
$5F = $E4 + $7B	transporte ativado

Se tomamos o estado do flag de transporte com indicação de que ocorreu um resultado negativo, poderemos interpretar $7B como um complemento de 2:

$7B em binário	= 01111011
Subtraindo 1	−1
Dá o complemento de 1	01111010
Negação	
Dá o complemento de 2	10000101 = 85

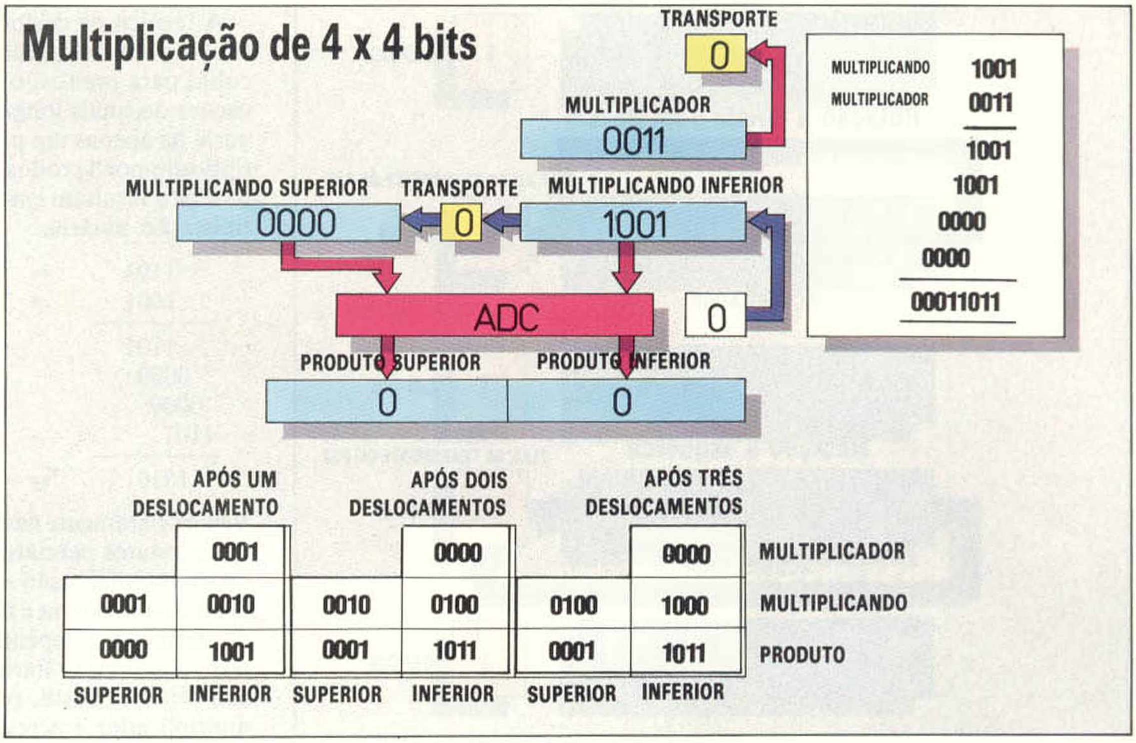
Deveríamos esperar então que $5F − $E4 resultasse no número negativo −$85. Verifiquemos esse resultado em notação decimal:

$5F =	95 em decimal
−$E4 =	288 em decimal
$85 =	−133 em decimal

Até aqui, portanto, tudo isso faz sentido. Suponhamos agora que a subtração em questão tenha sido uma soma de 2 bytes: $375F − $21E4. Vamos efetuá-la byte a byte:

Superior	Inferior	
$37	5F	= 14.175 em decimal
−$21	E4	= −8.676 em decimal
$15	7B	= 5.499 em decimal

Efetuando a subtração do byte inferior em primeiro lugar, o resultado será $7B e um transporte. Soma-se então esse transporte a $21, pela ins-



Deslocamentos na multiplicação
Este exemplo mostra uma multiplicação de 4 bits para maior clareza, pois o número de bits não afeta o algoritmo. Vê-se que o produto se forma pela adição de 0 ou de versões deslocadas do multiplicando, dependendo do valor de cada bit do multiplicador: 0 ou 1. Os bits do multiplicador deslocam-se para a direita através do flag de transporte, enquanto os bits do multiplicando deslocam-se para a esquerda, a partir do byte inferior até o byte superior, através do flag de transporte.

Operação de deslocamento
Usam-se as instruções de deslocamento e de rotação para examinar o conteúdo de um registrador, bit a bit. A cada deslocamento, o bit superior ou o inferior do registrador é deslocado para o flag de transporte do PSR; o estado desse flag serve para que uma instrução de desvio dirija o fluxo de controle do programa. As instruções de rotação agem de modo a preservar o conteúdo do registrador, mas as instruções de deslocamento lógico colocam 0 no registrador à medida que deslocam bits para fora. Desse modo, um deslocamento para a esquerda multiplica por dois o conteúdo do registrador, e um deslocamento para a direita divide-o por dois.

trução SBC, resultando $22, que é então subtraído de $37, dando $15. A resposta, $157B, está correta, como prova a versão em decimal.

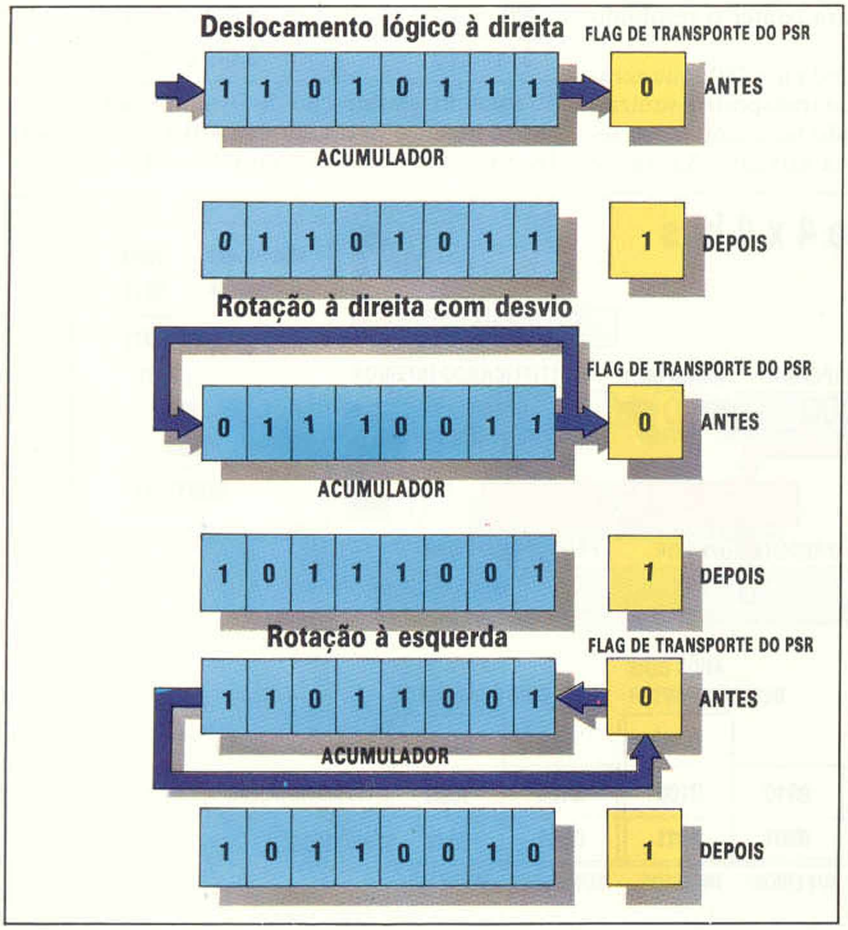
Como vemos, a aritmética de 2 bytes no Z80 segue estas regras simples:

1) Zerar o flag de transporte;
2) Subtrair os bytes inferiores, com transporte;
3) Subtrair os bytes superiores, com transporte.

A versão dessa seqüência no 6502 difere quanto ao primeiro aspecto — deve-se ativar, e não zerar, o flag de transporte, para que os bytes inferiores "emprestem" do byte superior. Não havendo empréstimo, a subtração prossegue normalmente, e o flag de transporte permanece ativado para subtração dos bytes superiores, o que ocorre de modo semelhante. Entretanto, se ocorrer um estouro negativo na subtração do byte inferior, o flag de transporte atuará como o "nono bit" do acumulador. Isso assegura um resultado correto e a desativação do flag de transporte. Quando se subtraem os bytes superiores com o flag de transporte desativado, o efeito é o mesmo que na subtração do byte superior no Z80, com o flag de transporte ativado — decrementa-se o número a ser subtraído antes que a operação ocorra. Ambas as maneiras de lidar com o empréstimo na subtração equivalem aos antigos métodos aritméticos de "emprestar" daqui e "devolver" ali. Examinemos mais detalhadamente a versão do 6502.

Se zerarmos o flag de transporte e subtrairmos $E4 de $5F, o resultado no acumulador será $7A e o flag de transporte permanecerá em zero. Vimos, pelo exemplo do Z80, que o resultado "verdadeiro" é $7B, com o flag de transporte indicando um número negativo. O valor $7B é o complemento de 2 da resposta "real" (–$85). Vemos que $7A é o complemento de 1 desse número e que o flag de transporte funciona como um interruptor para o modo do acumulador: ele é ativado para o complemento de 2, e zerado para o complemento de 1.

Se efetuarmos a subtração no 6502 com o flag de transporte ativado, o acumulador conterá $7B, e o flag de transporte será zerado. Se a subtração for de 2 bytes, zerar o flag de transporte vai decrementar o resultado da subtração do byte superior, compensando assim o "empréstimo" dos bytes inferiores.



Multiplicação

Vejamos como se efetua a soma na multiplicação decimal:

174	multiplicando
x209	multiplicador
1566	1.° produto parcial
000	2.° produto parcial
+348	3.° produto parcial
36366	produto final

Não é necessário entender a notação posicional para utilizar esse método; basta seguir procedimentos simples, efetuando multiplicações de um único dígito. A idéia básica nesse método é colocar cada produto parcial numa posição à esquerda do produto anterior. Uma vez admitida a necessidade desse posicionamento, a formação dos produtos parciais exige apenas o conhecimento da tabuada de multiplicação.

A técnica de deslocamento de produtos parciais e a memorização da tabuada é o que dificulta, para muitos, o aprendizado das multiplicações decimais longas. Já na multiplicação binária há apenas um produto real, que é um multiplicado por 1; todos os outros produtos de um só dígito resultam em 0. Observe esta longa multiplicação binária:

1101	=	13 em decimal
1001	=	9 em decimal
1101		1.° produto parcial
0000		2.° produto parcial
0000		3.° produto parcial
1101		4.° produto parcial
1110101	=	117 em decimal

Vemos claramente nesse exemplo o deslocamento dos produtos parciais, assim como a simplicidade da multiplicação em notação binária. Cada produto parcial ou é igual a 0 ou ao multiplicando deslocado, dependendo de o bit multiplicador valer 0 ou 1. Para executar uma multiplicação binária, basta, pois, examinar cada bit do multiplicador e acrescentar 0 ao total (se o bit

for 0) ou o multiplicando deslocado (se o bit for 1). Como, então, devemos proceder na linguagem ASSEMBLY para examinar cada bit do multiplicador e para deslocar o multiplicando?

Tanto no Z80 como no 6502, testamos o estado de um bit específico num byte por meio da instrução BIT. No Z80, essa instrução toma como operandos um endereço e um número de bit, ativando o flag zero se esse bit for 0 e desativando-o se for 1. No 6502, o operando é um endereço. O conteúdo desse endereço combina-se logicamente em E com o acumulador, e o flag zero é ativado ou desativado, dependendo de o resultado ser falso ou verdadeiro. Nenhum dos métodos á adequado aqui. Seria bem mais conveniente se o bit em questão funcionasse como flag de transporte ou flag zero, de modo que o fluxo do programa desviasse automaticamente de acordo com o estado de cada bit. Ambos os processadores realizam isso por instruções shift (deslocar), que também resolvem o problema de deslocar o multiplicando.

Há uma série de instruções de deslocamento e também de rotação em ambos os processadores, embora as do Z80 sejam mais complexas que as do 6502. De modo geral, elas deslocam cada bit de um registrador uma posição para a direita ou para a esquerda. Elas diferem quanto ao tratamento dos bits das extremidades do registrador, onde um bit é deslocado para fora, numa extremidade, enquanto outro vai para dentro, na outra extremidade. Se o bit 7 for deslocado para fora do registrador e logo recolocado no bit 0, a operação será de rotação para a esquerda. Se for o contrário, o 0 deslocado para o 7, a operação consistirá numa rotação para a direita. Num caso desses, o conteúdo do registrador muda de ordem, mas não se introduzem valores novos; e, após oito dessas rotações, o registrador volta ao estado original.

Não se empregando a rotação, será necessário um ponto de chegada para o bit deslocado para fora e outro para o bit deslocado para dentro. Ambos são supridos pelos vários flags de condição do PSR (Processor Status Register, "registrador de estado do processador") e, em especial, pelo flag de transporte. Ao criar uma sub-rotina para multiplicar dois números de um só byte, precisamos deslocar o multiplicando para a esquerda e o multiplicador para a direita. Os bits deste devem deslocar-se para fora, indo para o byte superior do multiplicando, enquanto se deslocam zeros para os bits não ocupados. Os bits multiplicadores devem deslocar-se através de um flag do PSR para teste, mas seu ponto de chegada e o estado dos que se deslocaram para dentro não é importante, a menos que seja necessário preservar o conteúdo do multiplicador. Com relação ao multiplicador, interessa-nos apenas saber se o bit que sai é 1 ou 0.

Assim, dado que o multiplicador armazena-se no endereço MPR, o multiplicando em MPDLO e o produto em PRODLO e PRODHI, podemos desenvolver assim as rotinas para multiplicação:

MULTIPLICAÇÃO DE 8 BITS

6502			Z80		
	ORG	$C100		ORG	$D000
START	LDA	#$00	START	LD	BC,(MPR)
	STA	PRODLO		LD	B,$08
	STA	PRODHI		LD	DE,(MPDLO)
	STA	MPDHI		LD	D,$00
	LDX	#8		LD	HL,$00
	CLC		LOOPO	SRL	C
LOOPO	ROR	MPR		JR	NC,CONTO
	BCC	CONTO		CALL	ADDIT
	JSR	ADDIT	CONTO	SLA	E
CONTO	ASL	MPDLO	ENDLPO	DJNZ	LOOPO
	ROL	MPDHI		LD	PRODLO
	DEX			RTS	
ENDLPO	BNE	LOOPO	MPR	DB	$E2
	RTS		MPDLO	DB	$7A
MPR	DB	$E2	MPDHI	DB	$00
MPDLO	DB	$7A	PRODLO	DW	$0000
MPDHI	DB	$00	ADDIT	ADD	HL,DE
PRODLO	DB	$00		RET	
PRODHI	DB	$00			
ADDIT	CLC				
	LDA	PRODLO			
	ADC	MPDLO			
	STA	PRODLO			
	LDA	PRODHI			
	ADC	MPDHI			
	STA	PRODHI			
	RTS				

Como mostra esse exemplo, a programação no Z80 é muito simplificada devido a seus registros de 16 bits e respectivas instruções. Em especial, compare a sub-rotina ADDIT nos dois programas. Na versão 6502, ROR gira o multiplicador para a direita, através do transporte; e ASL e ROL deslocam o multiplicando em direção à esquerda, para fora de MPDLO e para dentro de MPDHI, também através do transporte. O loop é controlado pelo registrador x agindo como contador.

Na versão do Z80, SRL desloca o multiplicador para a direita, através do transporte; e SLA e RI, o multiplicando para a esquerda, em DE, através do transporte. O loop é controlado pelo registrador B como um contador. Observe que a instrução ADD admite a aritmética de 16 bits, e não sofre influência do flag de transporte — ao contrário do que ocorre com ADC.

No próximo artigo, analisaremos os métodos de divisão e examinaremos vários métodos de controlar a apresentação na tela. Isso completará nosso curso.

Exercícios

A) Desenvolver uma sub-rotina de multiplicação usando um multiplicando de 16 bits e um multiplicador de 8 a sua escolha.

B) A multiplicação não passa de uma adição repetida: desenvolva uma sub-rotina de multiplicação de 8 x 8 bits, que não utilize as instruções de deslocamento nem de rotação.

TANDY MODEL 100

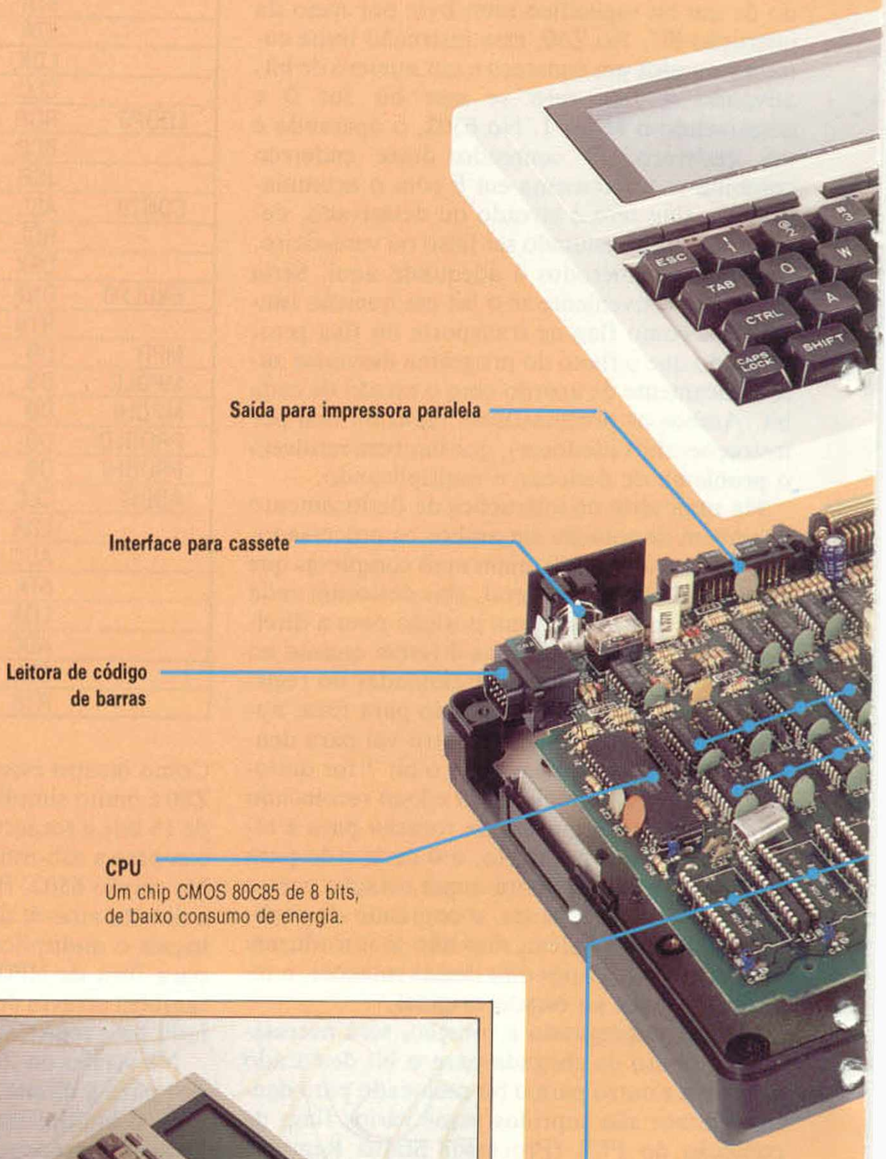
Quem viaja a negócios precisa de um computador portátil. Vamos ver como trabalham três desses pesos-pena, essencialmente semelhantes, com destaque para o Tandy Model 100.

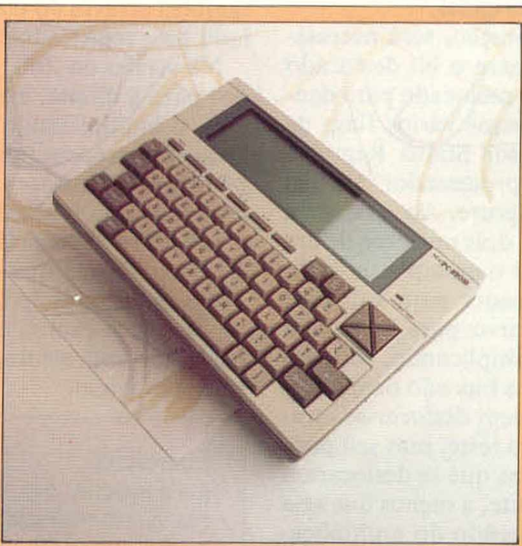
Um fabricante compra um produto acabado, muda a marca, a embalagem, alguns elementos, e vende-o como exclusivo. Esse processo é conhecido como OEM (Original Equipment Manufacture, "engenharia de griffe"). Técnica há muito aplicada em produtos como televisores e equipamentos de alta fidelidade, agora vem sendo usada no mercado de computadores. Três populares micros portáteis — o Tandy Model 100, o NEC PC-8201A e o Olivetti M10 — resultaram de um acordo desse tipo. Todos são fabricados pela mesma companhia, a indústria japonesa Kyocera, e vendidos para aquelas três empresas, que os acondicionam e comercializam com suas próprias marcas. Examinaremos em mais detalhe o Tandy Model 100 e ressaltaremos as diferenças entre ele e seus dois irmãos.

Pesando pouco menos de 1,8 kg, os três modelos sem dúvida classificam-se como portáteis. O Model 100 tem um teclado completo, tipo QWERTY, software residente em ROM incorporado e um LCD (visor de cristal líquido) acionado a bateria. O conteúdo da memória não se perde quando se desliga a máquina. Os arquivos ficam armazenados em RAM e são acessados diretamente, tal como ocorre em fita cassete ou disco. Preferindo-se armazenamento externo, o Model 100 também se conecta a um gravador cassete ou a uma unidade de disco, mas a memória permanente facilita a entrada de dados importantes quando se está em viagem.

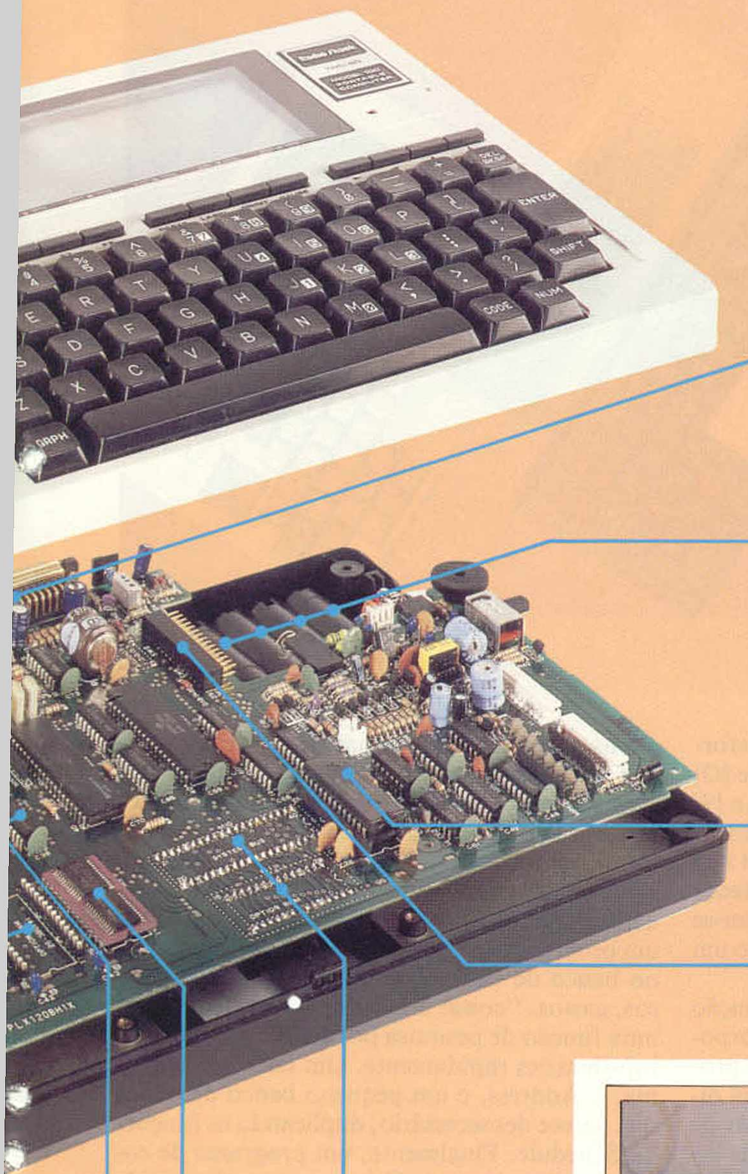
O LCD apresenta oito linhas de quarenta caracteres, mesclando textos e desenhos. A tela compõe-se de 15.360 pontos, endereçáveis um a um. Formam-se os caracteres, maiúsculos ou minúsculos, em matrizes de 6 x 8. O Model 100 dispõe de um conjunto completo de caracteres para diversas línguas, bem como de um conjunto especial de caracteres gráficos; já a máquina da NEC possui apenas três caracteres gráficos. Os LDCs do Tandy e do NEC são fixos; já o do Olivetti pode ser inclinado, proporcionando melhor ajuste ao ângulo de visão do operador. Para compensar a ausência desse recurso, os visores dos micros da NEC e da Tandy têm controles de contraste para melhorar a visibilidade.

O teclado do Tandy é de alta qualidade e apresenta teclas especiais para acessar os elementos gráficos incorporados. Outro recurso consiste na





NEC PC-8201A
Este modelo tem as mesmas dimensões dos concorrentes mas o lay-out de seu teclado é bem diferente. As teclas do cursor agrupam-se num pequeno tablete, e as de função reduzem-se de 8 para 5. Além disso, o NEC vem equipado com apenas três programas ROM: Text, Schedule e Telecom.



Saída para modem
Saída serial padrão RS232. O software para comunicações é residente na máquina.

Fonte de alimentação
O Model 100 funciona até 20 horas com quatro pilhas alcalinas 'AA' (tipo lapiseira). A memória interna mantém-se por até trinta dias, por meio de baterias de níquel-cádmio, recarregadas automaticamente quando se liga a máquina.

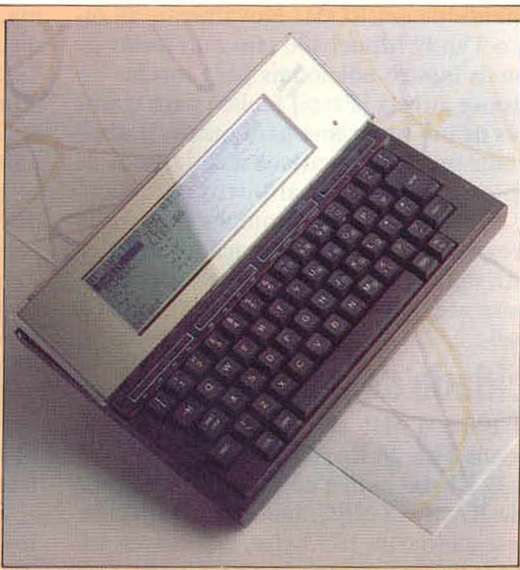
ROM padrão
Este chip contém o BASIC da Microsoft e os programas residentes.

Saída para o LCD
Um cabo multivias conecta o visor de cristal líquido à placa do sistema.

Bus para o sistema e conexões para ROM adicional
Slots vagos para expansão futura da ROM do sistema e para controle de entrada e saída.

RAM padrão de 8 k Kbytes

Unidade de teclado
Estes chips, que contêm os conjuntos de caracteres, controlam a entrada e a saída do teclado.



Olivetti M10
A versão Olivetti vem equipada com cinco programas residentes em ROM, além de apresentar um útil detalhe exclusivo: o visor de cristal líquido poder ser inclinado cerca de 40 graus, melhorando a visibilidade da tela sob diferentes condições de iluminação. O teclado é basicamente igual ao do Tandy.

TANDY MODEL 100

DIMENSÕES
300 x 215 x 50 mm.

CPU
CMOS 80C85 de 8 bits.

CLOCK
2,4 MHz.

MEMÓRIA
8 Kbytes ou 24 Kbytes de RAM, expansível em módulos de 8 Kbytes até um total de 32 Kbytes; ROM de 32 Kbytes, incluindo software e BASIC da Microsoft.

VÍDEO
LDC de 40 colunas x 8 linhas; gráficos de 240 x 64 pixels endereçáveis; caracteres ASCII e internacionais; 39 caracteres gráficos.

INTERFACES
Impressora paralela, fita cassete, saída serial RS232, leitora de código de barras, bus para o sistema.

LINGUAGENS DISPONÍVEIS
BASIC da Microsoft.

TECLADO
Tipo QWERTY, com 56 teclas; bloco numérico incorporado; oito teclas de função programáveis; quatro de comando e quatro de movimento do cursor.

DOCUMENTAÇÃO
Guia de referência rápida do BASIC, com 48 páginas; detalhado manual de operação, com duzentas páginas.

DIFERENÇAS
Olivetti M10:
57 teclas; 47 caracteres gráficos; tela inclinável; manual do usuário.
NEC PC-8201A:
57 teclas, sendo cinco de função; as do cursor vêm agrupadas; RAM de 16 Kbytes, expansível para 96 Kbytes; apenas três caracteres gráficos.

criação de um teclado numérico pela transformação das letras [M], [J], [K], [L], [U], [I] e [O] nos números de [0] a [6]; as teclas [7], [8] e [9] mantêm suas funções normais. Para a movimentação do cursor, os modelos Tandy e Olivetti têm quatro teclas pequenas, colocadas lado a lado, acima e à direita do teclado; no NEC, acham-se dispostas de modo mais funcional, formando um quadrado.

As três máquinas apresentam teclas de função programáveis, usadas com o software incorporado para manipulação de arquivos e dos programas mantidos em ROM. Novamente há diferenças: o Tandy Model 100 tem oito teclas de função, mais quatro adicionais para tarefas internas. Usa-se a tecla [Paste] (colar) para transferir dados de um programa para outro; [Label] para atribuir nomes específicos a cada tecla de função; [Print] para enviar arquivos para a impressora; e [Break] para interromper a execução de um programa. Essa configuração é idêntica no Olivetti, mas o NEC tem cinco teclas programáveis para até dez funções e também uma tecla [Pause].

Capacidade de memória

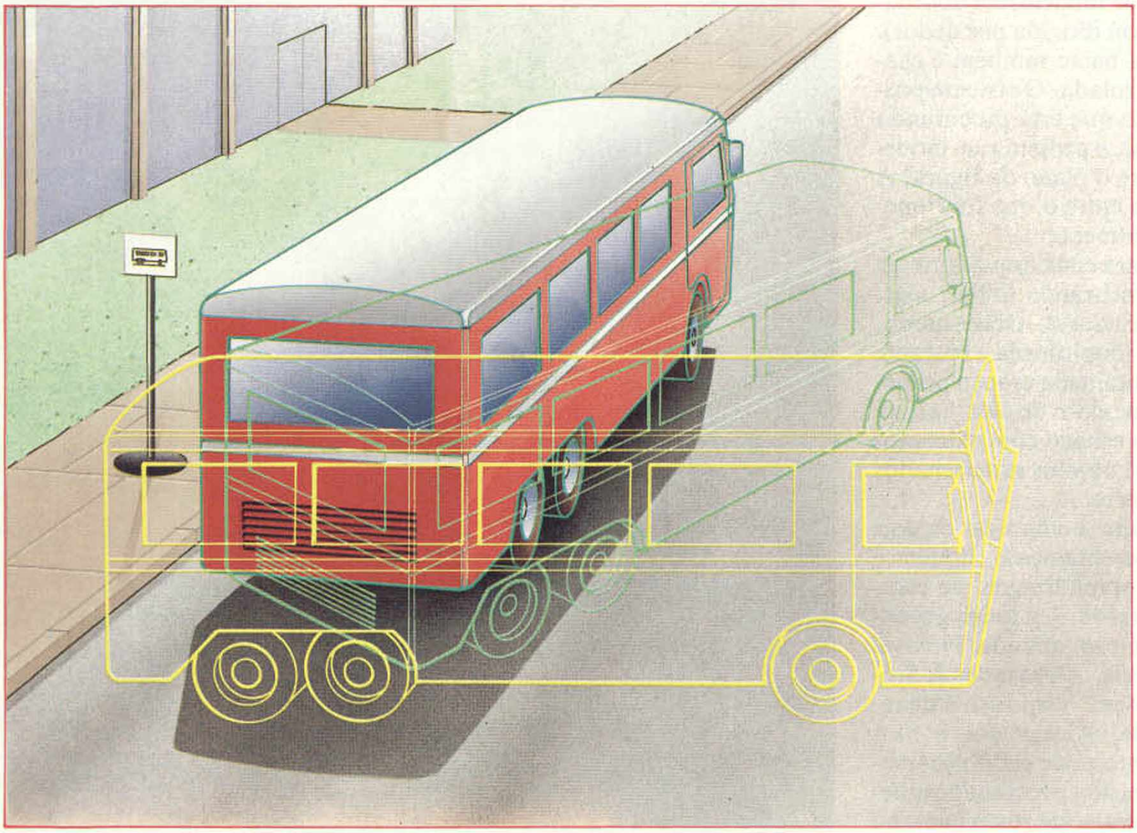
O Model 100 e o M10 são fornecidos em duas versões: com 8 ou 24 Kbytes de RAM, expansíveis para 32 Kbytes com a adição de um módulo interno de RAM. O NEC tem maior capacidade: é apresentado com 16 Kbytes, expansíveis para 64 Kbytes internamente, ou a 96 Kbytes, acoplados ao conector incorporado.

O Model 100 vem com o BASIC da Microsoft e um pequeno sistema de gerenciamento do software interno. Ao ser ligada, a máquina lista na tela arquivos armazenados na memória, bem como os nomes dos programas residentes fornecidos com o equipamento. Um deles é o Text, um pequeno processador de textos, adequado para escrever cartas, memorandos ou pequenos relatórios; especialmente apropriado para anotações rápidas, devendo ser de grande valia para jornalistas, estudantes ou homens de negócios. Há um programa de agenda, o Schedule, um pequeno banco de dados que armazena compromissos, gastos, "coisas a fazer" e outros lembretes; uma função de pesquisa possibilita encontrar as informações rapidamente. Um terceiro programa, o Address, é um pequeno banco de dados que parece desnecessário, duplicando as funções do Schedule. Finalmente, um programa de comunicações chamado Telecom controla a saída RS232 e permite a conexão do Model 100 a um modem e a linhas telefônicas — pressionando-se algumas teclas, enviam-se ou recebem-se dados de computadores distantes. O micro da NEC vem equipado apenas com BASIC, o Text e o Telecom.

As três máquinas são bem providas de interfaces: têm saída para comunicações RS232, saída para impressora paralela, interface para gravador cassete e entrada para leitora de código de barras. Os modelos Tandy e Olivetti incluem um bus de sistema, enquanto o NEC tem duas saídas seriais adicionais.

Com pequenas diferenças entre os três modelos, a utilização de uma máquina básica significa que os fabricantes podem fornecer um produto de alta qualidade, sem que uma só companhia tenha de arcar com todo o custo de desenvolvimento. A um preço relativamente acessível para as versões básicas, esses três portáteis oferecem um bom desempenho.

RECONHECIMENTO VISUAL



Uma boa adequação
Alguns sistemas de reconhecimento do padrão adotam abordagem de cima para baixo, na qual uma área ou cena é examinada à procura de determinado objeto. Uma representação deste, segundo as linhas que delineiam seu contorno (representação "em arame"), é oferecida numa tela, em vários ângulos, até que, se o objeto estiver presente, seja encontrada a projeção que se ajusta ao esboço da figura real. Na cena tridimensional apresentada ao lado, são inúmeras as projeções possíveis "em arame" antes de se encontrar a correta.

Ver e compreender o que se passa no mundo real é uma das principais metas da inteligência artificial. E já existem sistemas capazes de reconhecer padrões — e de mudar seu desempenho conforme o que "vêem".

Como se costuma dizer, ''A beleza está no olho do observador''; mas também se poderia afirmar que a beleza está no cérebro de quem a percebe — mais precisamente, numa complexa cadeia de processos neurais que começa no fundo da retina em alguma parte do córtex occipital do cérebro. É essa cadeia de eventos que os fisiologistas tentam entender e, até certo ponto, os especialistas em robôs procuram duplicar.

A percepção visual é um componente a tal ponto importante para nossa captação do ambiente que, com freqüência, ''ver'' é usado como sinônimo de perceber, compreender. De fato, a compreensão constitui a chave ao desafio da visão computadorizada. De algum modo, o computador deve obter informações, fornecidas por uma câmara ou outro dispositivo fotossensível, e compreender o que elas lhe dizem sobre o ambiente. Adquirir a informação é fácil, o problema está em como interpretá-la.

O processo de transformação de imagens em significados envolve três estágios principais: processamento da figura, reconhecimento do padrão e compreensão da imagem.

Processamento da figura. Confere nitidez a uma imagem borrada ou distorcida. (Problema relativamente fácil de resolver.)

Reconhecimento do padrão. Detecta a presença ou ausência de características ou objetos significativos. (Essa é uma tarefa de solução mais difícil.)

Compreensão da imagem. Entende o que se passa no mundo real. (Um problema ultracomplexo.)

O terceiro — a verdadeira visão computadorizada — ainda não foi alcançado. Contudo, já se conseguiram resultados úteis a partir dos estágios iniciais.

Reconhecimento do padrão

Vamos ''ver'' (entender) como funciona o segundo estágio. Em síntese, os reconhecedores de padrão classificam as imagens por comparação a um conjunto limitado de alternativas — por exemplo, as letras do alfabeto, no caso de sistemas envolvendo caracteres ópticos. Para isso, examinam partes da imagem digitalizada: pequenos grupos de pixels, os pontos e os grupos de pontos que compõem a imagem. O sistema pode então reconhecer e classificar padrões com base na presença ou ausência de certas características diferenciáveis.

Até agora foram desenvolvidas duas abordagens contrastantes da visão computadorizada na compreensão da imagem: a abordagem de cima para baixo (ou dirigida por modelo) e a abordagem de baixo para cima (ou dirigida por dados).

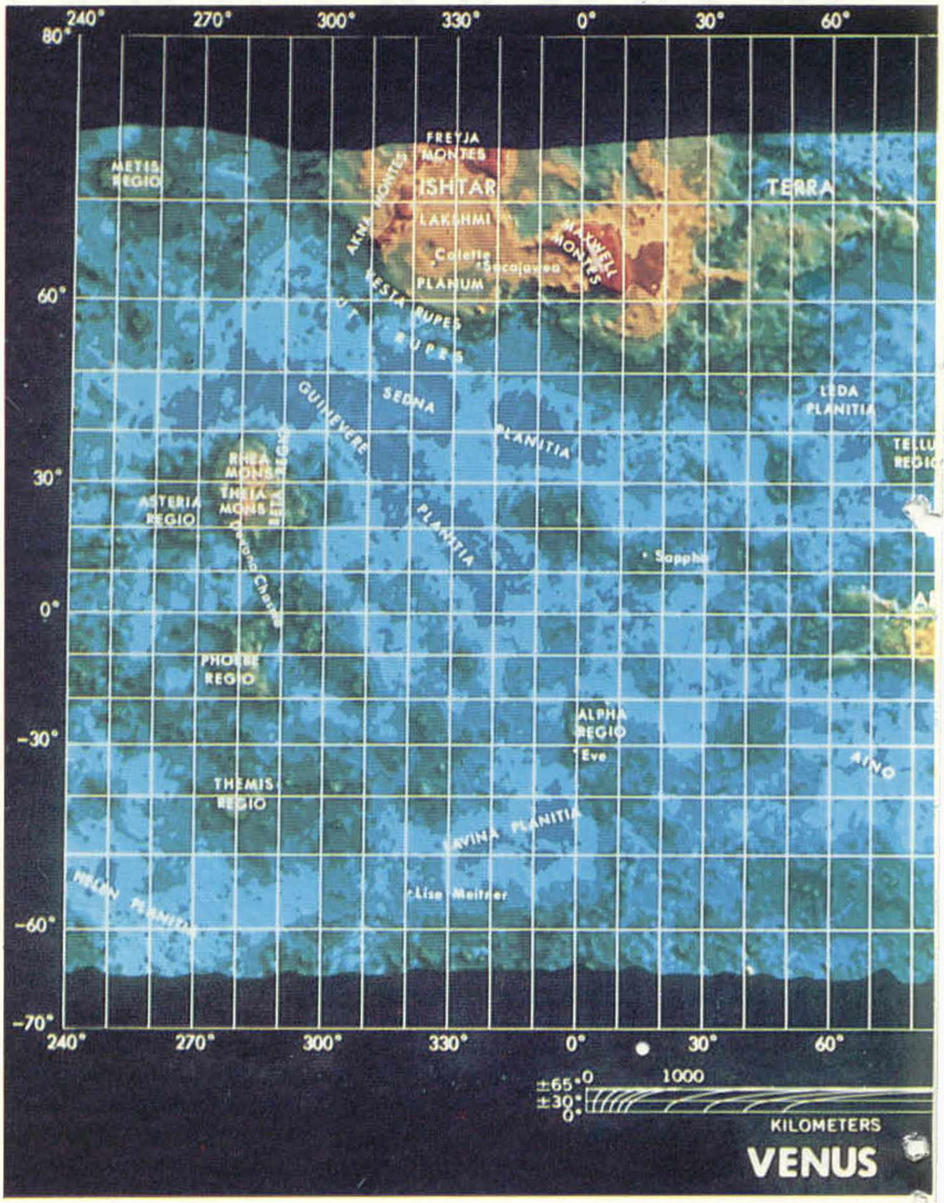
A análise de cima para baixo também é chamada de alucinação controlada. O sistema possui um modelo interno do que está procurando (um ônibus, por exemplo), e projeta esse modelo em vários ângulos sobre o plano da figura. A seguir, testa a adequação entre o que foi "imaginado" e o que está realmente lá.

Os sistemas de baixo para cima esquadrinham os dados da imagem, procurando linhas, margens e outros sinais reveladores. Desse modo, elaboram uma descrição simplificada e bastante abstrata da imagem, denominada esboço primário. A idéia é que, despojando-o de detalhes, será mais fácil comparar o esboço com exemplos de uma dada categoria de objetos armazenados como padrões de referência.

Sistemas práticos tendem a empregar os dois métodos, embora, até recentemente, houvesse bem poucos sistemas de aprendizagem no campo da visão computadorizada — e todos dependentes de inteligência pré-programada. Notável exceção é o Wisard (Wilkie, Aleksander & Stonham's Recognition Device, "dispositivo de reconhecimento de Wilkie, Aleksander e Stonham"). Esse equipamento pode ser programado para estabelecer distinção envolvendo sutilezas como, por exemplo, entre um rosto humano sorridente e outro carrancudo. Pode até mesmo generalizar esse conhecimento estendendo-se a rostos nunca antes "vistos". O Wisard opera em tempo real (25 quadros por segundo); seu sucesso comercial na classificação de chocolates em esteiras transportadoras pode abrir caminho para uma nova geração de "máquinas que enxergam", capazes de adaptar e aperfeiçoar seu desempenho.

Em linhas gerais, o Wisard trabalha alocando grupos de 8 pixels de cada vez (óctuplos) em bancos de RAM. Um óctuplo é um detector de características e pode estar em um de 256 estados (de 0 a 255), dependendo do estado de cada um dos pixels que ele monitora. Durante a fase de treinamento, um 1 é armazenado na localização de memória especificada pelo estado do óctuplo quando determinada imagem está presente. Mais tarde, na fase de reconhecimento, a presença de um 1 naquela localização evidencia que a imagem "ensinada" está presente mais uma vez.

Usando-se grande número de óctuplos, ou discriminadores, o sistema se torna relativamente impenetrável a dados irrelevantes (imagens com "ruído"). O sistema de Aleksander possui uma rede de 512 x 512 para a imagem e acima de 32.000 óctuplos, o que requer cerca de 8 milhões de bits (1 Mbyte) de RAM. Só há pouco tempo tais capacidades de memória se tornaram economicamente viáveis.

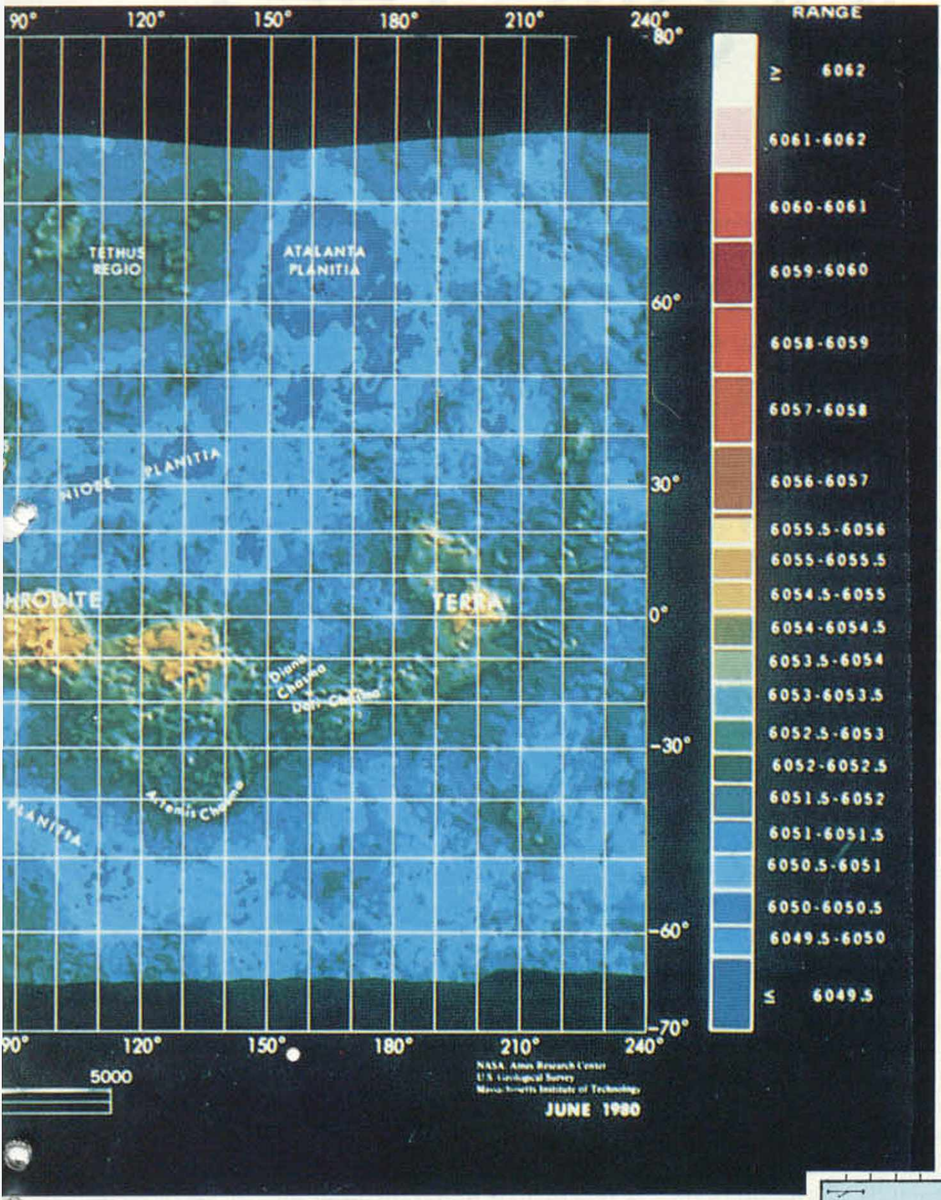


0	0	0	0	0	0	0	0
0	0	0	1	1	0	0	0
0	0	1	0	1	1	0	0
0	1	1	0	0	0	1	0
0	0	1	1	1	1	0	0
0	0	1	0	0	1	0	0
0	1	1	0	0	1	0	0
0	1	1	0	0	0	1	0

0	0	0	0	0	0	0	0
0	0	0	1	1	1	0	0
0	0	1	0	0	1	0	0
0	1	0	0	0	1	0	0
0	1	1	1	1	0	0	0
0	1	1	0	0	1	0	0
0	1	0	0	0	1	0	0
0	1	0	0	0	0	1	0

A diferença entre R e A

Acima estão formas digitalizadas das letras A e R, mas nenhum dos padrões é bem definido: ambos apresentam manchas e distorções. Dados com esse tipo de "ruído" são um problema constante em todos os sistemas de inteligência artificial. Será que você conseguiria identificar as letras?



Processamento de figura

Esta foto de Vênus, obtida por uma sonda espacial, dá um exemplo de processamento de figuras. O computador acentuou os contornos entre áreas de diferentes altitudes. Tudo isso implica o uso de "cores falsas".

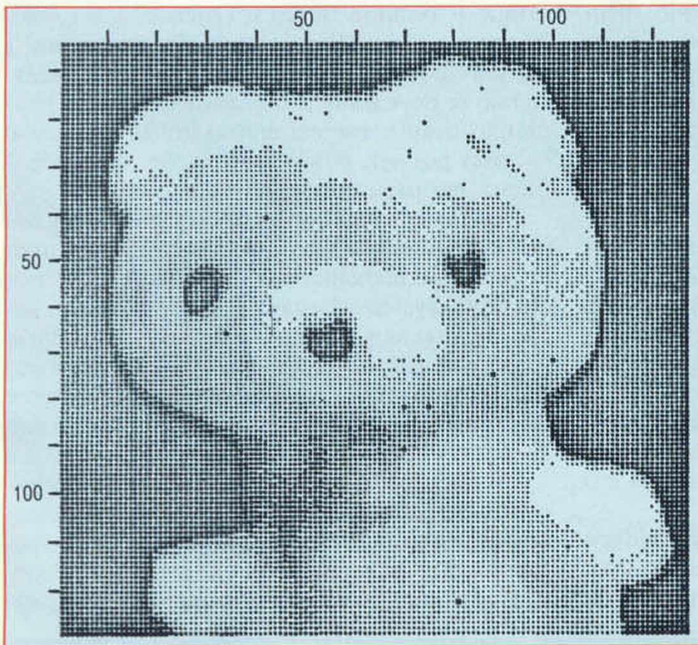
Na verdade, jamais poderíamos ver essa imagem sem a ajuda do computador, pois Vênus está sempre envolto em nuvens espessas, e nossos olhos não são sensíveis aos comprimentos de onda necessários para atravessá-las. Tais figuras mostraram-se muito úteis (às vezes, muito bonitas); baseiam-se nos comprimentos de onda do rádio, infravermelho e ultravioleta, e também da luz visível.

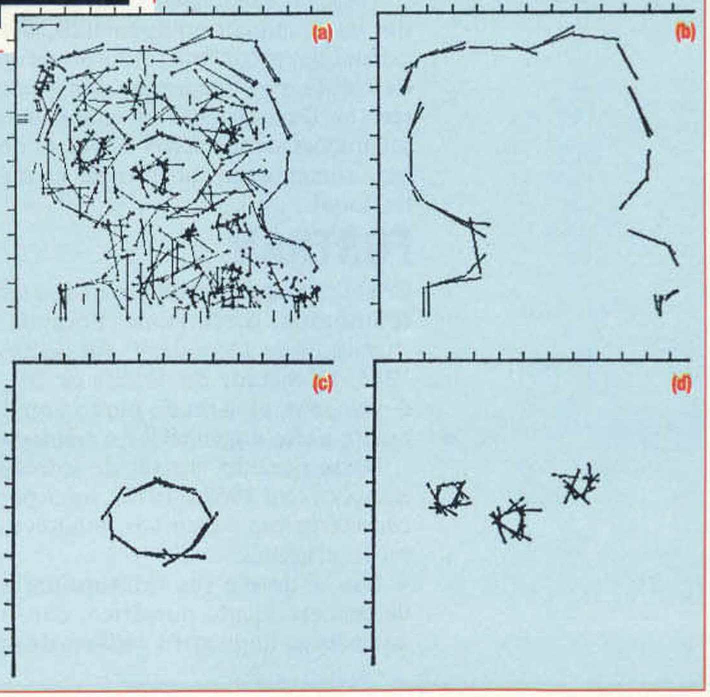
O leito oceânico também é mapeado pelo sonar, com o qual se obtêm "imagens acústicas", que se baseiam na sondagem por ultra-som, sem qualquer relação com as radiações eletromagnéticas (luz visível, infravermelho, ultravioleta etc.).

Outras modalidades de processamento de figuras são utilizadas para "limpar" imagens borradas e distorcidas. O uso desses algoritmos já se tornou rotineiro.

Esboços primários

Existe um método de processamento visual que não conta com o conhecimento prévio do que o padrão pretende representar. O esboço primário (a) extrai os contornos e a forma primitiva da imagem do ursinho de brinquedo por comparação das regiões adjacentes. Extratos adicionais do esboço primário — de (b) até (d) — exibem grupos importantes de elementos, que auxiliam no reconhecimento da imagem original.





BABEL INFORMÁTICA

As diversas linguagens de programação — desde as desenvolvidas na década de 50 e ainda em uso até as mais recentes — permitem ampla gama de opções para o processamento de quase toda espécie de dados.

Já foi demonstrado matematicamente que qualquer linguagem de computador pode simular o modo de operação de outras linguagens. Portanto, em termos dos problemas que podem resolver, todas elas se equivalem. Mas algumas se mostram muito mais adequadas para a resolução de determinado tipo de problema, enquanto outras possibilitam o desenvolvimento de programas mais eficientes.

Das inúmeras linguagens existentes a maior parte se volta para finalidades bem específicas — como o C, para a programação de sistemas, e o PROLOG, para a programação lógica. As chamadas linguagens genéricas não exigem do programador o controle dos recursos do equipamento (vale dizer: são "processuais", especificando apenas as operações que devem ser efetuadas pela máquina); constituem exemplos o FORTRAN, o COBOL, o APL etc.

Excluímos desta breve descrição as linguagens já tratadas com maior detalhe no decorrer da série, como o BASIC, o PASCAL e o ASSEMBLY.

Todas as linguagens consideradas podem rodar em qualquer equipamento, desde que este possua um programa tradutor apropriado, além de recursos — memória, por exemplo — suficientes. Deve-se levar em conta ainda que implementações para micros de várias dessas linguagens somente são encontradas no mercado internacional.

FORTRAN

Primeira linguagem de alto nível a ser amplamente utilizada, o FORTRAN (Formula Translator, "tradutor de fórmulas") foi desenvolvido pela IBM, na metade da década de 50. Trinta anos depois, continua sendo muito popular, especialmente entre engenheiros e cientistas.

Nesse período, apesar de sofrer duas padronizações (em 1962 e 1977), incorporando novas características e recursos, manteve a forma e o estilo originais.

Isso se deve à sua extraordinária capacidade de processamento numérico, constituindo uma das poucas linguagens capazes de manipular diretamente números complexos. Quase sempre compilado, o FORTRAN possui símbolos e regras sintáticas bastante semelhantes às convenções da matemática e das linguagens naturais (o inglês e o português, por exemplo). Os vários programas existentes em FORTRAN podem ser usados e adaptados para qualquer finalidade científica ou técnica; daí ser improvável que ele venha a se tornar obsoleto.

A linguagem dispõe de recursos como loops, ramificações, sub-rotinas, funções e atribuição de valores a nomes variáveis. Permite um número limitado de tipos de dados: números inteiros, números reais de precisão simples e dupla, números complexos, expressões booleanas e strings. Por outro lado, a única estrutura disponível para a organização dos dados é a tabela. As variáveis aceitam nomes de até seis caracteres, começando com uma letra. A menos que seja especificado de outro modo, todos os nomes de variáveis que comecem com as letras de I a N representam números inteiros.

No FORTRAN, um programa, uma sub-rotina ou uma definição de função sempre incluem uma instrução de cabeçalho (para identificação), uma série de instruções de especificação (para definição dos tipos de dados e declaração das dimensões da tabela), e, por fim, uma lista de comandos de execução (que constituem o corpo principal do programa).

COBOL

Existem mais linhas de programação escritas em COBOL (Common Business Oriented Language, "linguagem comum para atividades comerciais") do que em todas as outras linguagens juntas. Isso não se deve à sua prolixidade — mesmo programas simples exigem muitas linhas em COBOL —, mas por ser, desde sua criação, a principal linguagem para aplicações comerciais.

Rigidamente padronizado, o COBOL tem seu uso e desenvolvimento supervisionado por uma comissão permanente, a CODASYL (Conference On Data Systems Languages, "conferência sobre linguagens e sistemas de dados"). A própria linguagem foi projetada, em 1959, por essa comissão, que reuniu representantes do governo e dos principais fabricantes de computadores dos Estados Unidos.

Criado especialmente para aplicações comerciais e de gerenciamento, o COBOL caracteriza-se por programas compostos de palavras e frases, em linguagem natural, que podem ser decodificadas com facilidade por usuários sem qual-

quer formação técnica. A grande força da linguagem reside nos poderosos recursos para manipulação de arquivos, responsável pelo seu absoluto domínio na área empresarial.

O desenvolvimento e a operação de um programa em COBOL mostram diferenças em relação às outras linguagens, embora possamos encontrar muitos dos conceitos existentes em todas as linguagens processuais. Os programas apresentam-se em quatro divisões, conhecidas pelos nomes identification, environment, data e procedure division.

A identification division inclui os nomes do programa e do programador, as datas em que foi escrito e compilado, bem como os comentários introdutórios. A environment division foi planejada inicialmente para ser dependente da máquina, especificando detalhes do computador no qual o programa foi escrito e executado. Muitos desses detalhes encontram-se hoje sob controle do sistema operacional, e o principal uso dessa divisão é a especificação dos arquivos externos usados no programa. A data division contém detalhes de todas as áreas de dados usadas pelo programa. Ela compreende duas seções principais: uma que abriga a disposição dos registros para os arquivos externos, e outra para os dados usados apenas num programa.

As operações e procedimentos a executar com os dados são especificados na procedure division, que, se necessário, pode ser dividida em seções. Cada uma destas, ou toda a divisão, compõe-se de parágrafos formados por sentenças. Uma sentença pode corresponder a um ou mais comandos. A estrutura da procedure division foi projetada para ficar tão próxima quanto possível do inglês. Isso, juntamente com os identificadores que chegam a ter trinta caracteres, torna os programas compreensíveis até para leigos em programação.

ALGOL

Resultado de um programa internacional para a elaboração de uma linguagem algorítmica padronizada, o ALGOL (Algorithmic Language, "linguagem algorítmica") teve sua primeira versão apresentada em 1958. Contudo, ainda sofreu várias reformulações e aperfeiçoamentos até o lançamento da especificação oficial, em 1960.

Trata-se de uma linguagem de aplicação genérica. Bastante poderosa, permite a expressão precisa de procedimentos numéricos. Caracteriza-se por utilizar uma notação algébrica completa e tem como finalidade a solução de problemas lógicos por meio de algoritmos elegantes e eficientes. Embora semelhante ao FORTRAN, o ALGOL dispõe de maiores recursos e possui uma estrutura extremamente formalizada, que possibilita a eliminação de exceções e maior legibilidade.

Apesar de não ser tão famoso quanto o FORTRAN e o COBOL, ocupa uma posição destacada na evolução das linguagens de programação. Foi a partir de uma versão do ALGOL que Niklaus Wirth desenvolveu, em 1968, a linguagem PASCAL.

Um dos motivos que impediram a popularização do ALGOL é o amplo uso que ele faz de símbolos pouco comuns, além da complexidade cada vez maior que foi adquirindo de uma versão para outra. Suas implementações restringem-se a pesquisas universitárias, especialmente na Europa.

LISP

Desenvolvida no período de 1956 a 1958, a linguagem LISP (List Processor, "processador de listas") não apenas continua em uso, como vem se tornando cada vez mais importante nas pesquisas de inteligência artificial. Isso em função de sua impressionante capacidade para manipular listas.

O LISP foi projetado especificamente para processar dados não numéricos cujo comprimento e estrutura, muitas vezes, variam durante o processamento. Nessa linguagem, define-se uma lista como um conjunto de itens em determinada ordem entre parênteses. Por isso, o LISP oferece um modo conveniente para a representação de dados, como frases em linguagens naturais, fórmulas matemáticas, teoremas lógicos ou até mesmo programas de computadores. Esse é um dos conceitos mais poderosos do LISP, e tanto os dados como os programas são representados do mesmo modo. Isso facilita muito a criação de programas que desenvolvem e rodam outros programas.

Um programa em LISP nada mais é do que uma coleção de funções, escrita em forma de lista. Quase tudo nessa linguagem se faz por meio de funções, o que a torna especialmente apta para aplicações, tais como: geração e verificação de provas matemáticas, reconhecimento de padrões, processamento algébrico, simulação de formas humanas de resolução de problemas, programação heurística, análise lingüística e exploração e desenvolvimento de novas linguagens de programação não processuais.

Outra característica poderosa do LISP é a sua extensibilidade. Significa que ele permite ao programador acrescentar recursos que se tornam parte integrante da linguagem.

PL/1

O PL/1 (Programming Language 1, "linguagem de programação 1") foi lançado em 1965 como uma linguagem de programação genérica, voltada para a resolução de problemas tanto comerciais como científicos. Combinando recursos do FORTRAN (por exemplo, instruções simples e concisas) e a capacidade de manipulação de arquivos do COBOL, o PL/1 possui uma eficiência somente comparável à sua complexidade.

A linguagem se baseia no uso de estruturas chamadas procedimentos — blocos de instrução que executam as funções desejadas. Um determinado procedimento pode ser incorporado hierarquicamente a outro, e quaisquer dados declarados ficam automaticamente disponíveis para todos os procedimentos subsidiários. O PL/1 dispõe de recursos para a manipulação de strings alfanuméricos ou strings de bits. Além disso, permite que os programadores descrevam os dados em termos de tabelas e outras estruturas "apontadoras".

Apesar de sua impressionante versatilidade, a linguagem PL/1 não encontrou muita aceitação. A complexidade foi um dos empecilhos à sua difusão: a maioria das implementações emprega mais de duzentas palavras-chaves. Além disso, apenas uma pequena parte de seus recursos é utilizada na maioria das aplicações, o que certamente desestimulou os usuários que já conheciam o FORTRAN e o COBOL.

APL

Elaborado por Kenneth Iverson, em 1967, a partir de uma notação especial para o processamento de equações diferenciais aplicadas a modelos econômicos, o APL (A Programming Language, "uma linguagem de programação") logo se tornou motivo de muita polêmica.

Muitos o consideram a linguagem processual mais flexível, poderosa e concisa que existe. Mas seus adversários criticam-lhe a sintaxe e a pouca adequação ao processamento de tarefas "produtivas" e de programas que envolvam muita edição.

Utilizando uma notação muito semelhante à da matemática e operando em modo interativo, o usuário do APL com algum treino em ciências exatas não encontra dificuldade para obter resultados com seus programas iniciais.

Dentre as linguagens processuais de aplicação genérica, o APL distingue-se por possuir o mais amplo conjunto de operadores primitivos. Por meio deles, o programador pode executar funções como: geração de números aleatórios, cálculo fatorial, formação e inversão de matrizes etc. Esses operadores, acionados por uma única tecla, realizam tarefas que em outras linguagens exigiriam dezenas de comandos.

O APL é muito popular entre matemáticos e estatísticos devido à velocidade com que os algoritmos podem ser desenvolvidos e testados. No entanto, ele não se restringe às aplicações numéricas, sendo também usado numa ampla gama de aplicações comerciais, tais como produção de documentos, análises de gráficos e análises financeiras. Além disso, mostra-se particularmente apropriado para trabalhos de pesquisa e aplicações educacionais.

PROLOG

O PROLOG (Programming Logic, "lógica de programação") começou a ficar conhecido depois que os japoneses o escolheram para o desenvolvimento dos computadores inteligentes de quinta geração. Constitui, ao lado do LISP, uma das ferramentas básicas para as pesquisas de inteligência artificial.

Linguagem recente, o PROLOG foi inventado no início da década de 70 pelo francês Alain Colmerauer, da Universidade de Marselha. Baseia-se na lógica dos predicados, semelhante à que usamos na vida cotidiana para a resolução de problemas, embora possua notação específica e determinadas regras. Define-se um predicado como uma relação entre coisas. Na frase "João ama Maria", o predicado é "ama". Em PROLOG, isso seria escrito da seguinte maneira: "ama (João, Maria)". Embora a leitura seja mais difícil, essa notação evidencia o predicado e seus argumentos.

Por meio da lógica dos predicados, pode-se descrever o mundo em termos de fatos e implicações. E, a partir desses fatos, deduzir outros. No PROLOG, isso é feito por intermédio das variáveis lógicas, que se assemelham àquelas existentes no BASIC e em algumas outras linguagens de programação.

Foi visando proporcionar ao programador um modo de descrever a estrutura lógica de um programa por meio de um banco de dados (com fa-

tos) e regras lógicas que se projetou o PROLOG. Sua sintaxe é bastante simples, mas a terminologia muito confusa. O elemento mais importante — o termo — pode ser uma constante, uma variável ou uma estrutura. As estruturas são criadas a partir de constantes e variáveis. Uma estrutura muito comum nos programas em PROLOG, conhecida como fato, compõe-se de um predicado isolado ou seguido por uma lista de argumentos.

O PROLOG, embora seja considerado uma linguagem não processual (o programador só precisa descrever o problema, pois a linguagem faz o resto), possui mecanismos rígidos para resolução de problemas, os quais não são facilmente modificados.

De qualquer modo, trata-se de um código particularmente adequado ao processamento de linguagens naturais. E, como essa área é um aspecto essencial das verdadeiras máquinas user-friendly ("amigável ao usuário"), fundamental para as pesquisas de computadores de quinta geração, o PROLOG tem seu lugar assegurado no desenvolvimento tecnológico do final do século XX.

C

Linguagem de programação genérica, o C não foi projetado para resolver um tipo específico de problema. Não se trata, portanto, de uma linguagem de alto nível, como o BASIC, o PASCAL e outras; nem de uma linguagem de baixo nível, como o ASSEMBLY.

Os programadores de sistemas são os principais usuários do C, que reúne as vantagens das linguagens de alto e de baixo nível. Além disso, ele não se limita a um sistema operacional determinado, possibilitando a utilização de seus programas em computadores que usam os mais diversos processadores.

Em virtude da feroz competição para lançar novos produtos no mercado, essa versatilidade se mostra fundamental para as indústrias de microcomputadores, em especial quanto ao desenvolvimento de software para os novos modelos de 16 bits. Afinal, um programa recém-desenvolvido precisa ser rapidamente adaptado aos quatro ou cinco principais microprocessadores. Com a linguagem C, apenas aqueles trechos do programa que usam código de máquina específico precisam ser reescritos, mantendo-se inalterada a maior parte do programa. Por isso, o C vem atraindo cada vez mais atenção dos principais fabricantes de software.

Tal como algumas linguagens de alto nível, o C foi projetado no sentido de permitir uma programação estruturada. Empregando blocos e nomes flexíveis para programas e variáveis, facilita a elaboração, leitura e depuração. Ao mesmo tempo, o usuário tem à sua disposição dezenas de operadores de baixo nível, que correspondem exatamente às operações em linguagem de máquina de vários microprocessadores. Com o uso de tais comandos, uma linha em código C pode substituir várias linhas em código de máquina.

Outra vantagem do C está no fato de seus programas serem executados com muito mais rapidez do que a possibilitada por qualquer linguagem de alto nível; além de ser mais facilmente aprendida do que o ASSEMBLY.

Código extremamente compacto, atua com cerca de trinta palavras reservadas. Todas as versões são escritas na própria linguagem (e não em ASSEMBLY), principal motivo de sua transportabilidade e facilidade de implementação em outros sistemas.

Com regras bastante flexíveis, em geral C permite ao programador fazer o que quiser, desde que não haja proibição explícita. A linguagem foi desenvolvida de forma a pressupor um programador profissional. Por exemplo, ela permite a manipulação de bits individuais; os apontadores têm acesso direto às posições de memória, sendo possível incluir ou definir novas estruturas de dados quando as disponíveis revelam-se insuficientes. Por outro lado, embora tais recursos impliquem considerável aumento de potência e flexibilidade dos programas escritos em C, eles também aumentam a possibilidade de erros obscuros difíceis de localizar e corrigir.

Desde que foi criado, em 1972, por Dennis Ritchie, da Bell Laboratories, o C tem sido associado ao sistema operacional Unix. Desenvolvidos originalmente para o minicomputador PDP 11, da Digital Equipment Corporation, tanto o C quanto o Unix foram implementados em todos os tipos de computadores, dos micros ao supercomputador Cray-1.

Por haver sido criado para avançados sistemas que operavam em modo compartilhado, o Unix revelou-se flexível e poderoso o suficiente para atender às necessidades dos programadores profissionais que trabalham no desenvolvimento de novas gerações de microcomputadores, como os de 16 e 32 bits. O Unix foi escrito em C, o que o torna transportável para uma grande variedade de microprocessadores.

MEDIDATA

Mesmo produzindo mínis, micros e supermicrocomputadores, a Medidata adota como filosofia de trabalho o assessoramento aos clientes. Sua preocupação é oferecer soluções "sob medida".

Criada em outubro de 1976, a Medidata no princípio atuava apenas como bureau de prestação de serviços. Isso era feito através de terminais interligando seus clientes à central de serviços, que já naquela época utilizava minicomputadores.

No ano seguinte ao de sua fundação, quando o governo brasileiro passou a enfocar a política nacional de informática de maneira mais dedicada e consistente, a companhia tratou logo de garantir uma fatia nesse mercado. Para tanto, uniu sua especialização em prestação de serviços ao desenvolvimento de tecnologia própria de software, para otimizar o desempenho de seus futuros equipamentos. Em 1978, assim que a empresa se viu reestruturada de forma segura, lançou seu primeiro computador, o sistema M2001.

Mesmo tendo ramificado as atividades, a filosofia de trabalho da Medidata continua sendo o atendimento direto aos clientes. Seu sistema de trabalho é conhecido na empresa como "lógica sob medida". Refere-se ao tratamento especializado no tocante à automação dos departamentos de cada empresa-cliente, em vista das diferentes necessidades de cada uma em função do tamanho, do ramo de atividade e da filosofia.

O M301
O microcomputador profissional da Medidata. Embora com 16 bits, opera também 8, ampliando sua faixa de utilização.

Seguindo à risca sua proposta de trabalho, a Medidata desenvolve projetos e softwares básicos, de apoio e aplicativos, fabrica computadores, presta serviços de consultoria e desenvolvimento de projetos especiais, além da prestação de serviços de apoio, manutenção e treinamento.

Para atender seus clientes, a companhia possui uma estrutura própria compatível com as necessidades de manutenção, assistência técnica e de suporte aos equipamentos por eles adquiridos. A empresa dispõe de engenheiros, técnicos e analistas distribuídos por uma rede de filiais e centros de atendimento. Para agilizar esse serviço foi criado o Telessuporte, onde o cliente recebe um diagnóstico e as medidas corretivas para seu equipamento através de linhas telefônicas e terminais remotos.

Os centros de treinamento estão equipados com recursos audiovisuais, salas de aula, professores e aparelhagem técnica. Neles os clientes da Medidata recebem instruções visando maior familiarização com os diversos produtos comercializados pela empresa. Usuários com um tipo de problema específico são atendidos por uma equipe voltada para o projeto e implementação de sistemas aplicativos. Esse grupo desenvolve soluções especiais, adequadas a cada necessidade. Entre os clientes já atendidos por essa modalidade de serviços incluem-se a Rede Globo de Televisão, a Esso Brasileira de Petróleo e o Bank of Boston.

Para atender a um mercado crescente, com uma demanda vertiginosa de novos produtos e serviços, a Medidata oferece uma série de programas aplicativos, básicos e de apoio. Esses pacotes foram desenvolvidos de forma integrada e modular, e cobrem as funções básicas de administração financeira, de materiais e de recursos humanos. Como as necessidades variam de acordo com a filosofia de trabalho de cada empresa, os aplicativos da Medidata podem ser modificados e ajustados pelo próprio usuário ou pela Medidata.

O Gerenciador de Aplicações, um programa de apoio desenvolvido pela empresa, permite acelerar o desenvolvimento de aplicações e aumentar a produtividade. O software básico é o MUMPS, que pode ser utilizado em todos os seus computadores, favorecendo a compatibilidade entre os vários aparelhos.

Na linha dos equipamentos de uso profissional, a Medidata fabrica mínis, micros e supermicros. O M301 é um microcomputador de 16 bits, compatível com o IBM PC e com o micro CP/M80 de 8 bits, que utiliza o sistema MUMPS. O multiusuário M1001, um supermicrocomputador, possibilita a interligação de até dez terminais, além de compartilhar impressoras e discos, e permitir comunicação com a rede nacional de telex. No terreno dos mínis, são produzidos o M2001, interligando até vinte terminais, locais ou remotos, com 384 Mbytes de capacidade em discos, e também o M3001, para até 64 terminais, locais ou remotos, e 768 Mbytes de memória em disco.

vice, "dispositivo de interface de som") do Commodore 64, faz bom uso dos três osciladores disponíveis. A impressão geral é de que as possibilidades do 64 são, no entanto, plenamente exploradas.

Os jogadores podem optar por competir em todos os eventos, praticar uma modalidade ou ver uma lista dos recordes mundiais de cada uma. Também há uma opção para uso de um ou dois joysticks, o que permite aos jogadores a disputa direta — um contra o outro ou contra o computador — nas provas de natação e de corridas.

A primeira das oito modalidades é o salto com vara, talvez a mais difícil de todas. Os jogadores participam um de cada vez, respondendo a uma série de sinais de orientação fornecidos pelo computador. Após a barra ser colocada na altura mínima de 4 m, pergunta-se ao jogador se ele deseja tentar seu salto. Se a resposta for afirmativa, o computador então pedirá que o jogador tome posição na pista. A seguir o atleta começa a correr pela tela.

A fim de apoiar a vara na caixa, o jogador deve puxar o joystick para trás. Empurrando-o para a frente, faz o atleta subir; e, pressionando o botão de disparo, passa-o por sobre a barra sem derrubá-la. Cada uma dessas ações requer uma sincronização de décimos de segundo — um erro de cálculo em qualquer ponto causará a queda da vara.

Essa seqüência é uma boa ilustração dos parâmetros a serem considerados na codificação. Além da necessidade de manter a tela todo o tempo, a fim de assegurar um desenvolvimento suave nos gráficos, o computador deve também checar os movimentos do joystick e do botão de disparo. Finalmente, precisa verificar se a vara e o atleta estão perpendiculares e posicionados para um bom salto. Durante esse processo, o usuário não deve notar que essas ações estão se realizando.

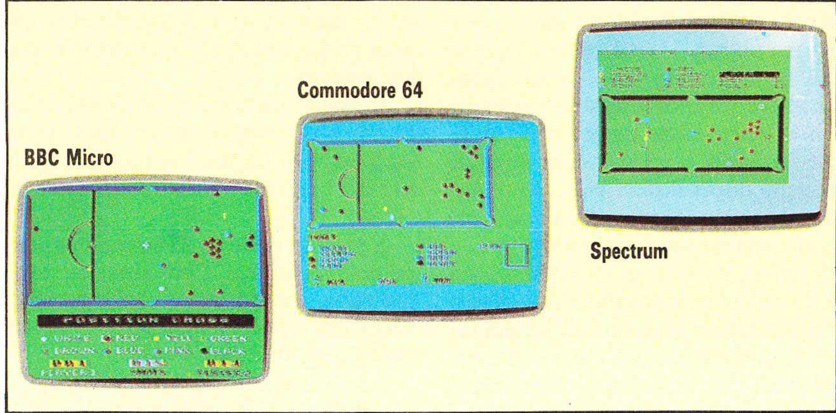
Depois de cada modalidade, uma tabela mostra as medalhas conquistadas, seguindo-se a execução do hino nacional do país vencedor. O computador então carrega a modalidade seguinte. O fato de cada uma ser carregada separadamente mostra quantas linhas de programação são necessárias para se rodar cada competição.

As modalidades que se seguem são os saltos ornamentais e a ginástica. Nelas, o jogador deve manipular o joystick para produzir um mergulho ou um salto elegante, certificando-se de terminar o movimento com uma entrada perfeita na água ou uma saída perfeita dos aparelhos. O computador avalia, então, seu desempenho.

Summer Games: Para os computadores Commodore 64, Atari e Apple II.
Editor: Epyx Software, Estados Unidos.
Autores: Randy Glover, Stephen Landrum, John Leupp, Brian McGhie, Stephen Mundry, Erin Murphy e Scott Nelson.
Joysticks: Necessários.
Formato: Versão em disco.

SETE NA CAÇAPA

Snooker (análise)



Sinuca de bico
A essência da sinuca está na precisão tanto da mirada como da força nas tacadas. No Snooker, da Visions, um cursor de mira deve ser posicionado na mesa, controlando-se a força da tacada pela pressão e liberação do botão de disparo.

Com o aparecimento da televisão em cores, no final da década de 60, o jogo de sinuca tornou-se um esporte apreciado pelos telespectadores ingleses e, nos anos 80, também pelos brasileiros. À primeira vista, a sinuca parece inviável para uma simulação em computador. No entanto, uma versão simplificada do jogo foi colocada no mercado para três microcomputadores.

Para os profissionais desse esporte, a idéia de computadorizar o jogo de sinuca representa um sacrilégio. Como — perguntam — seria possível quantificar e programar cada curva sutil de uma bola com efeito ou cada variação das características da mesa? Certamente não dentro dos limitados 16 Kbytes de um Spectrum!

Entretanto, a sinuca pode ser imaginada como uma aplicação bem simples do princípio da conservação de energia cinética — e a álgebra nela envolvida é facilmente tratada por um computador.

Quando se começa a jogar o Snooker, da Visions, aparece na tela uma vista superior da mesa, com as bolas representadas por círculos de cores diferentes. Joga-se com a bola branca. Para tanto, move-se uma cruz pelo vídeo até encontrar uma posição satisfatória para definir a trajetória da bola branca, que é então disparada na direção da bola do jogo. Determina-se a força pelo tempo em que se mantém pressionada a tecla ou o botão de disparo.

Depois de algumas tacadas, um jogador experiente percebe dois problemas importantes. Primeiro, que o computador — Commodore, Atari ou Apple II — não pode calcular com precisão os ângulos de impacto ou o atrito com o pano para obter resultados convincentes numa tela de televisão. Sig-

nifica que muitas tacadas dão resultados quase imprevisíveis (o que, convenhamos, também pode acontecer num jogo de verdade). Algumas são mesmo impossíveis para o computador. Para compensar, as caçapas são proporcionalmente maiores que o normal.

O outro problema se mostra na prática bem mais profundo: é que se vê a mesa em planta. Vale dizer que, quando se vai jogar, não se tem a vantagem de prever a trajetória olhando por trás do taco. Além disso, sendo o vídeo levemente curvo, a previsão dos ângulos apresenta dupla dificuldade. As tentativas iniciais do jogador, portanto, correm o risco de ser frustrantes.

Todavia, uma vez aceito que o jogo guarda pouca relação com a sinuca verdadeira e depois de dominadas algumas de suas peculiaridades, ele passa a ser fascinante por si mesmo, envolvendo refinamentos bem interessantes.

O jogador começa a notar algumas características sutis da programação; em particular, que é possível dar efeito à bola, embora os resultados dessa tacada sejam, uma vez mais, imprevisíveis.

As três versões diferem na maneira como refletem os pontos fortes e fracos das máquinas em que são jogadas. Surpreendentemente, o jogo emprega pouca memória e, portanto, é carregado com rapidez nas três máquinas — mesmo no Commodore, notório por sua baixa velocidade de carregamento. Em conseqüência, a qualidade de cada versão reflete as possibilidades gráficas de cada máquina, e não sua capacidade de memória.

O Spectrum mostra-se o mais fraco nesse aspecto. A mesa é pequena, e nem todas as bolas são apresentadas nas cores verdadeiras. A versão para o Commodore tem a mesa maior, cores realistas e bolas mais bem dimensionadas.

O marcador é mostrado com mais destaque nessa versão, e pode-se optar por uma demonstração no início do jogo — embora seja preciso completar uma partida antes de o controle voltar ao usuário.

A versão para o BBC é a melhor do grupo; sua mesa cobre virtualmente toda a largura da tela, sendo reforçada por um controle muito mais preciso do taco.

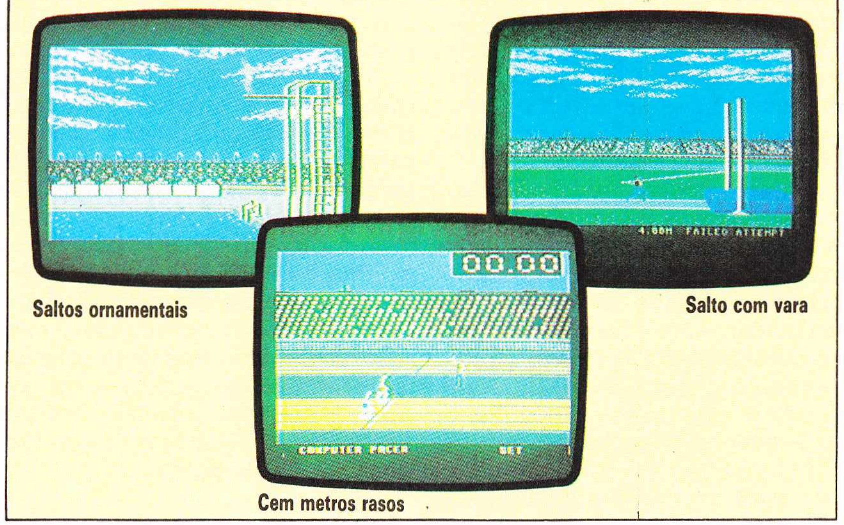
A partida pode ser disputada por um ou dois jogadores. Como muitos jogos para computadores são extremamente solitários, a partida para dois jogadores ganha pontos nesse quesito.

Infelizmente a Visions não viu oportunidade de dar um ritmo cadenciado e tranqüilo ao software, como no jogo real. Embora a impressão de atropelo prevaleça, pode-se imaginar que seu potencial — em função de tanta memória disponível — ainda não esteja totalmente desenvolvido.

Snooker: Para os computadores Commodore 64, Apple II e Atari.
Editor: Visions (Software Factor) Ltd., Inglaterra.
Autor: Tim Bell.
Joysticks: Opcionais.
Formato: Disco ou cassete.

COMPETIÇÕES ESPORTIVAS

Summer Games (análise)



Saltos ornamentais
Cem metros rasos
Salto com vara

Rumo ao pódio
As telas permitem avaliar a qualidade do Summer Games, da Epyx. Cada modalidade é carregada individualmente a partir de disco ou fita, o que possibilita armazenar e processar grande quantidade de dados em alta resolução para os fundos.

Os jogos olímpicos de Los Angeles, em 1984, despertaram grande interesse dos usuários por softwares baseados em esportes competitivos. A maior parte desses pacotes é produzida nos Estados Unidos, embora alguns — a exemplo do Summer Games, aqui examinado — sejam originários da Grã-Bretanha.

O pacote da empresa americana Epyx inclui oito modalidades esportivas — de atletismo e natação a ginástica e tiro. Cada uma pode ter até oito competidores diferentes, com direito a escolher um entre dezessete países para representar. As nações cujas bandeiras não foram incluídas no programa competem pelo estandarte da Epyx.

Uma vez carregado o jogo, a primeira cena mostra a cerimônia de abertura. O transporte da tocha ocorre com o acompanhamento de uma música-tema adequada ao evento. O atleta que a leva à tribuna acende a pira, e, nesse momento, solta-se um bando de pombos. O som e os gráficos da cerimônia de abertura são típicos do pacote como um todo. O fundo é cuidadosamente desenhado com gráficos mapeados de alta resolução, e os movimentos suaves do atleta e dos pombos resultam dos excelentes recursos de sprite do Commodore.

A música, ainda que não seja o melhor exemplo do que se pode conseguir com o chip SID (Sound Interface De-